S00P0272 US00

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: １９９９年 ３月 ４日

出 願 番 号
Application Number: 平成１１年特許願第０５７６８９号

出 願 人
Applicant (s): ソニー株式会社

CERTIFIED COPY OF
PRIORITY DOCUMENT

２０００年 ２月 ４日

特 許 庁 長 官
Commissioner,
Patent Office 近 藤 隆 彦

【書類名】　特許願

【整理番号】　9801164206

【提出日】　平成11年 3月 4日

【あて先】　特許庁長官殿

【国際特許分類】　G06F 15/00

【発明者】

【住所又は居所】　東京都品川区北品川６丁目７番３５号　ソニー株式会社内

【氏名】　岡村　英明

【特許出願人】

【識別番号】　000002185

【氏名又は名称】　ソニー株式会社

【代表者】　出井　伸之

【代理人】

【識別番号】　100067736

【弁理士】

【氏名又は名称】　小池　晃

【選任した代理人】

【識別番号】　100086335

【弁理士】

【氏名又は名称】　田村　榮一

【選任した代理人】

【識別番号】　100096677

【弁理士】

【氏名又は名称】　伊賀　誠司

【手数料の表示】

【予納台帳番号】　019530

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9707387

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　データ処理装置、データ処理方法及びプログラム提供媒体

【特許請求の範囲】

【請求項１】　メッセージ通信を行う複数のオブジェクトから構成されるオブジェクト指向オペレーティングシステムを実行するデータ処理装置において、

所定のオブジェクトの追加を要求する複合化要求メッセージを受信したオブジェクトが、当該所定のオブジェクトを構成オブジェクトとして参照するためのテーブルデータ構造を作成し、当該構成オブジェクトのデータによって初期化することによって複合オブジェクトを構成する手段と、

少なくとも一つの当該構成オブジェクトのデータ構造を作成し、当該テーブルデータ構造に登録し、当該構成オブジェクトが具備する少なくとも一つのメッセージ処理機能と当該メッセージ処理機能を要求するためのメッセージインターフェースとの関係を、当該構成オブジェクトデータ構造に登録する手段と、

を具備することを特徴とするデータ処理装置。

【請求項２】　上記複合オブジェクトは、上記テーブルデータ構造を初期化するために、上記構成オブジェクトの名前と、当該構成オブジェクトの有する上記メッセージインターフェースの数と、当該構成オブジェクトを初期化する処理機能とを記載した、所定の設定のデータ構造を読み込むこと

を特徴とする請求項１記載のデータ処理装置。

【請求項３】　上記複合オブジェクトは、独自の実行スレッドを持つことにより、上記構成オブジェクトに対して発行されたメッセージの処理を、当該実行スレッドにおいて実行すること

を特徴とする請求項１記載のデータ処理装置。

【請求項４】　上記複合オブジェクトは、所定の構成オブジェクトの追加要求を受理することにより、上記テーブルデータ構造に、当該所定の構成オブジェクトのデータ構造を追加登録すること

を特徴とする請求項３記載のデータ処理装置。

【請求項５】　上記複合オブジェクトに所定のオブジェクトを追加する際、当該複合オブジェクトを構成する全ての構成オブジェクトとの間に、実行逐次性が

あることを検査する手段と、

当該実行逐次性が確認された後、上記所定のオブジェクトを追加する手段と、

を具備すること特徴とする請求項４記載のデータ処理装置。

【請求項６】　上記実行逐次性は、

上記所定のオブジェクトにメッセージが送信された時点で、上記複合オブジェクトを構成する全ての構成オブジェクトが、当該所定のオブジェクトと並行に動作する必要がないことを検査する手段と、

当該所定のオブジェクトが、当該複合オブジェクトを構成するいずれかの構成オブジェクトに対しメッセージを送信する場合に、当該構成オブジェクトが既に他のメッセージを処理中であることはないことを検査する手段と、

当該複合オブジェクトを構成するいずれかの構成オブジェクトから、当該所定のオブジェクトがメッセージを受け取る場合に、当該所定のオブジェクトが既に他のメッセージを処理中であることはないことを検査する手段と、

によって検査されること

を特徴とする請求項５記載のデータ処理装置。

【請求項７】　上記構成オブジェクトは、その他全ての上記構成オブジェクトとの間で、実行逐次性があることを検査すること

を特徴とする請求項４記載のデータ処理装置。

【請求項８】　上記構成オブジェクトは、所定の構成オブジェクトの分離要求を受理することにより、上記テーブルデータ構造から、当該所定の構成オブジェクトのデータ構造を登録抹消すること

を特徴とする請求項３記載のデータ処理装置。

【請求項９】　上記複合オブジェクトは、

当該複合オブジェクトに送達されたメッセージの送付元が、非構成オブジェクトからのものであるか、当該構成オブジェクトからのものであるかを検査する手段と、

当該メッセージの送付先が、当該非構成オブジェクトに対するものであるか、当該構成オブジェクトに対するものであるかを検査する手段と、

を具備すること

を特徴とする請求項３記載のデータ処理装置。

【請求項１０】　上記複合オブジェクトは、

上記メッセージの送付元が上記構成オブジェクトであり、当該メッセージの送付先が当該構成オブジェクトに対するものである場合、実行スレッドの切り替えをせずに、メッセージで要求された処理を実行すること

を特徴とする請求項９記載のデータ処理装置。

【請求項１１】　メッセージ通信を行う複数のオブジェクトから構成されるオブジェクト指向オペレーティングシステムのデータ処理方法において、

所定のオブジェクトの追加を要求する複合化要求メッセージを受信したオブジェクトが、当該所定のオブジェクトを構成オブジェクトとして参照するためのテーブルデータ構造を作成し、当該構成オブジェクトのデータによって初期化することによって複合オブジェクトを構成するステップと、

少なくとも一つの当該構成オブジェクトのデータ構造を作成し、当該テーブルデータ構造に登録し、当該構成オブジェクトが具備する少なくとも一つのメッセージ処理機能と当該メッセージ処理機能を要求するためのメッセージインターフェースとの関係を、当該構成オブジェクトデータ構造に登録するステップと

を具備することを特徴とするデータ処理方法。

【請求項１２】　上記複合オブジェクトは、上記テーブルデータ構造を初期化するために、上記構成オブジェクトの名前と、当該構成オブジェクトの有する上記メッセージインターフェースの数と、当該構成オブジェクトを初期化する処理機能とを記載した、所定の設定のデータ構造を読み込むこと

を特徴とする請求項１１記載のデータ処理方法。

【請求項１３】　上記複合オブジェクトは、独自の実行スレッドを持つことにより、上記構成オブジェクトに対して発行されたメッセージの処理を、当該複合オブジェクトの実行スレッドにおいて実行すること

を特徴とする請求項１１記載のデータ処理方法。

【請求項１４】　上記複合オブジェクトは、所定の構成オブジェクトの追加要求を受理することにより、上記テーブルデータ構造に、当該所定の構成オブジェクトのデータ構造を追加登録すること

を特徴とする請求項１３記載のデータ処理方法。

【請求項１５】　上記複合オブジェクトは、所定のオブジェクトを追加する際、

当該複合オブジェクトを構成する全ての構成オブジェクトとの間に、実行逐次性があることを検査する検査ステップと、

当該実行逐次性が確認された後、上記所定のオブジェクトを追加する追加ステップとを実行すること

を特徴とする請求項１４記載のデータ処理方法。

【請求項１６】　上記実行逐次性は、

上記所定のオブジェクトにメッセージが送信された時点で、上記複合オブジェクトを構成する全ての構成オブジェクトが、当該所定のオブジェクトと並行に動作する必要がないことを検査するステップと、

当該所定のオブジェクトが、当該複合オブジェクトを構成するいずれかの構成オブジェクトに対しメッセージを送信する場合に、当該構成オブジェクトが既に他のメッセージを処理中であることはないことを検査するステップと、

当該複合オブジェクトを構成するいずれかの構成オブジェクトから、当該所定のオブジェクトがメッセージを受け取る場合に、当該所定のオブジェクトが既に他のメッセージを処理中であることはないことを検査するステップと、

によって検査されること

を特徴とする請求項１５記載のデータ処理方法。

【請求項１７】　上記構成オブジェクトは、その他全ての上記構成オブジェクトとの間で、実行逐次性があることを検査すること

を特徴とする請求項１４記載のデータ処理方法。

【請求項１８】　上記構成オブジェクトは、所定の構成オブジェクトの分離要求を受理することにより、上記テーブルデータ構造から、当該所定の構成オブジェクトのデータ構造を登録抹消すること

を特徴とする請求項１３記載のデータ処理方法。

【請求項１９】　上記複合オブジェクトは、

当該複合オブジェクトに送達されたメッセージの送付元が、非構成オブジェク

トからのものであるか、当該構成オブジェクトからのものであるかを検査するステップと、

当該メッセージの送付先が、当該非構成オブジェクトに対するものであるか、当該構成オブジェクトに対するものであるかを検査するステップと、

を具備すること

を特徴とする請求項１３記載のデータ処理方法。

【請求項２０】　上記複合オブジェクトは、

上記メッセージの送付元が上記構成オブジェクトであり、当該メッセージの送付先が当該構成オブジェクトに対するものである場合、実行スレッドの切り替えをせずに、メッセージで要求された処理を実行すること

を特徴とする請求項１９記載のデータ処理方法。

【請求項２１】　メッセージ通信を行う複数のオブジェクトから構成されるオブジェクト指向オペレーティングシステムのデータ処理プログラムを提供するプログラム提供媒体において、

所定のオブジェクトの追加を要求する複合化要求を受理したオブジェクトが、当該所定のオブジェクトを構成オブジェクトとして参照するためのテーブルデータ構造を作成し、当該構成オブジェクトのデータによって初期化することによって複合オブジェクトを構成するステップと、

少なくとも一つの当該構成オブジェクトのデータ構造を作成し、当該テーブルデータ構造に登録し、当該構成オブジェクトが具備する少なくとも一つのメッセージ処理機能と当該メッセージ処理機能を要求するためのメッセージインターフェースとの関係を、当該構成オブジェクトデータ構造に登録するステップと

を具備するデータ処理プログラムを提供するプログラム提供媒体。

【請求項２２】　オブジェクト指向オペレーティングシステムを実行するデータ処理装置において、

オブジェクト間でメッセージ通信を行うオブジェクトを、１つ以上の構成オブジェクトから構成される複合オブジェクトと、複合オブジェクト以外のオブジェクトである標準オブジェクトとのいずれかにより構成するオブジェクト構成手段と、

上記オブジェクト構成手段により構成された標準オブジェクト及び構成オブジェクトを任意のオブジェクトから参照できるように、各標準オブジェクト及び各構成オブジェクトに識別子を付す識別子設定手段と、

上記オブジェクト構成手段により構成されたオブジェクトのうち、複合オブジェクトについては、１つの複合オブジェクトを１つの実行スレッドによって実行し、その実行スレッドを複合オブジェクトを構成する各構成オブジェクトによって共有させる実行スレッド制御手段と

を備えること特徴とするデータ処理装置。

【請求項２３】　所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加することを要求するメッセージが入力されたときに、少なくとも、上記所定のオブジェクトを特定する追加対象オブジェクト名と、当該所定のオブジェクトを構成オブジェクトとして他のオブジェクトに追加するのに必要な初期化手続きが記述されたメソッドを特定する初期化メソッド情報とを読み込む追加オブジェクト情報読み込み手段と、

上記追加オブジェクト情報読み込み手段により読み込まれた上記追加対象オブジェクト名により、他のオブジェクトに構成オブジェクトとして追加するオブジェクトを特定する追加オブジェクト特定手段と、

上記追加オブジェクト特定手段により特定されたオブジェクトを、上記追加オブジェクト情報読み込み手段により読み込まれた上記初期化メソッド情報によって特定されるメソッドを実行することにより、構成オブジェクトとして他のオブジェクトに追加するオブジェクト追加手段と、

を備えることを特徴とする請求項２２記載のデータ処理装置。

【請求項２４】　所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加する際に、構成オブジェクトとして追加されるオブジェクトに関する情報が格納される記述子を上記識別子と対応づけて作成する記述子作成手段と、

上記記述子作成手段によって作成された記述子に、少なくとも、構成オブジェクトとして追加されるオブジェクトが具備するメソッドを呼び出すための情報を格納するメソッド情報格納手段と、

を備えることを特徴とする請求項２３記載のデータ処理装置。

【請求項２５】　所定の構成オブジェクトを複合オブジェクトから削除することを要求するメッセージが入力されたときに、少なくとも、削除対象の構成オブジェクトを特定する削除対象オブジェクト名を読み込む削除オブジェクト情報読み込み手段と、

上記削除オブジェクト情報読み込み手段により読み込んだ削除対象オブジェクト名により、複合オブジェクトから削除する構成オブジェクトを特定する削除オブジェクト特定手段と、

上記削除オブジェクト特定手段により特定された構成オブジェクトを複合オブジェクトから削除するオブジェクト削除手段と、

上記削除オブジェクト特定手段により特定された構成オブジェクトに対応した識別子から、当該構成オブジェクトに対応した記述子を特定し、当該記述子を削除する記述子削除手段と、

を備えることを特徴とする請求項２４記載のデータ処理装置。

【請求項２６】　上記オブジェクト構成手段は、

複合オブジェクトを複数の構成オブジェクトから構成する際に、

ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージが送信された時点で、それらの２つの構成オブジェクトが並行動作する必要がないという条件と、

ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージを送信するときに、メッセージを受け取る側の構成オブジェクトが他のメッセージを処理中であることはないという条件と

を満たすように複合オブジェクトを構成すること

を特徴とする請求項２２記載のデータ処理装置。

【請求項２７】　あるオブジェクトから他のオブジェクトにメッセージを送るときに、メッセージ送信側オブジェクト及びメッセージ受信側オブジェクトが構成オブジェクトであり、それらの構成オブジェクトが同じ複合オブジェクトに含まれている場合、上記実行スレッド制御手段は、実行スレッドの切り替えを行わずに、メッセージ送信側オブジェクトが使用していた実行スレッドと同じ実行ス

レッドを用いて、メッセージ送信側オブジェクトから送られたメッセージで要求された処理を、メッセージ受信側オブジェクトにより実行させること

を特徴とする請求項２２記載のデータ処理装置。

【請求項２８】　オブジェクト間でのメッセージ通信に使用されるアプリケーションプログラムインターフェースとして、メッセージ通信を行うオブジェクトが標準オブジェクトであるか構成オブジェクトであるかに関わらず共通に使用可能なアプリケーションプログラムインターフェースを備えること

を特徴とする請求項２２記載のデータ処理装置。

【請求項２９】　オブジェクト指向オペレーティングシステムによって実行されるデータ処理方法において、

オブジェクト間でメッセージ通信を行うオブジェクトを、１つ以上の構成オブジェクトから構成される複合オブジェクトと、複合オブジェクト以外のオブジェクトである標準オブジェクトとのいずれかにより構成するとともに、

任意のオブジェクトから標準オブジェクト及び構成オブジェクトを参照できるように、各標準オブジェクト及び各構成オブジェクトに識別子を付し、

複合オブジェクトについては、１つの複合オブジェクトを１つの実行スレッドによって実行し、その実行スレッドを複合オブジェクトを構成する各構成オブジェクトによって共有させること

を特徴とするデータ処理方法。

【請求項３０】　所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加することを要求するメッセージが入力されたとき、

少なくとも、上記所定のオブジェクトを特定する追加対象オブジェクト名と、当該所定のオブジェクトを構成オブジェクトとして他のオブジェクトに追加するのに必要な初期化手続きが記述されたメソッドを特定する初期化メソッド情報とを読み込み、

上記追加対象オブジェクト名により、他のオブジェクトに構成オブジェクトとして追加するオブジェクトを特定し、

上記追加対象オブジェクト名により特定されたオブジェクトを、上記初期化メソッド情報によって特定されるメソッドを実行することにより、構成オブジェク

トとして他のオブジェクトに追加すること

を特徴とする請求項２９記載のデータ処理方法。

【請求項３１】　所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加する際に、

構成オブジェクトとして追加されるオブジェクトに関する情報が格納される記述子を上記識別子と対応づけて作成し、

上記記述子に少なくとも、構成オブジェクトとして追加されるオブジェクトが具備するメソッドを呼び出すための情報を格納すること

を特徴とする請求項３０記載のデータ処理方法。

【請求項３２】　所定の構成オブジェクトを複合オブジェクトから削除することを要求するメッセージが入力されたとき、

少なくとも、削除対象の構成オブジェクトを特定する削除対象オブジェクト名を読み込み、

上記削除対象オブジェクト名により、複合オブジェクトから削除する構成オブジェクトを特定し、

上記削除対象オブジェクト名により特定された構成オブジェクトを複合オブジェクトから削除するとともに、

上記削除対象オブジェクト名により特定された構成オブジェクトに対応した識別子から、当該構成オブジェクトに対応した記述子を特定し、当該記述子も削除すること

を特徴とする請求項３１記載のデータ処理方法。

【請求項３３】　複合オブジェクトを複数の構成オブジェクトから構成する際、

ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージが送信された時点で、それらの２つの構成オブジェクトが並行動作する必要がないという条件と、

ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージを送信するときに、メッセージを受け取る側の構成オブジェクトが他のメッセージを処理中であることはないという

条件と

を満たすように複合オブジェクトを構成すること

を特徴とする請求項２９記載のデータ処理方法。

【請求項３４】　あるオブジェクトから他のオブジェクトにメッセージを送るときに、メッセージ送信側オブジェクト及びメッセージ受信側オブジェクトが構成オブジェクトであり、それらの構成オブジェクトが同じ複合オブジェクトに含まれている場合は、

実行スレッドの切り替えを行わずに、メッセージ送信側オブジェクトが使用していた実行スレッドと同じ実行スレッドを用いて、メッセージ送信側オブジェクトから送られたメッセージで要求された処理を、メッセージ受信側オブジェクトにより実行すること

を特徴とする請求項２９記載のデータ処理方法。

【請求項３５】　オブジェクト間でメッセージ通信を行う際に、メッセージ通信を行うオブジェクトが標準オブジェクトであるか構成オブジェクトであるかに関わらず共通に使用可能なアプリケーションプログラムインターフェースを用いること

を特徴とする請求項２９記載のデータ処理方法。

【請求項３６】　オブジェクト指向オペレーティングシステムのデータ処理プログラムを提供するプログラム提供媒体であって、

上記オブジェクト指向オペレーティングシステムのデータ処理プログラムは、

オブジェクト間でメッセージ通信を行うオブジェクトを、１つ以上の構成オブジェクトから構成される複合オブジェクトと、複合オブジェクト以外のオブジェクトである標準オブジェクトとのいずれかにより構成するとともに、

任意のオブジェクトから標準オブジェクト及び構成オブジェクトを参照できるように、各標準オブジェクト及び各構成オブジェクトに識別子を付し、

複合オブジェクトについては、１つの複合オブジェクトを１つの実行スレッドによって実行し、その実行スレッドを複合オブジェクトを構成する各構成オブジェクトによって共有させること

を特徴とするプログラム提供媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、オブジェクト指向オペレーティングシステムを実行するデータ処理装置、オブジェクト指向オペレーティングシステムでのデータ処理方法、並びに、オブジェクト指向オペレーティングシステムのデータ処理プログラムを提供するプログラム提供媒体に関する。

【０００２】

【従来の技術】

オブジェクト指向技術をオペレーティングシステムに適用し、オペレーティングシステムの構成要素をオブジェクトとしてモジュール化する技術がある。このように、構成要素がオブジェクトとしてモジュール化されて構成されたオペレーティングシステムは、オブジェクト指向オペレーティングシステムと称される。なお、オペレーティングシステム上で実行されるアプリケーションオブジェクトと、オペレーティングシステムの構成要素となるオブジェクトとが同様な実行機構を持つオブジェクト指向オペレーティングシステムは、純オブジェクト指向オペレーティングシステムと称される場合もある。

【０００３】

オブジェクト指向オペレーティングシステムにおいて、オペレーティングシステムの提供するサービスは、オブジェクトの集合により定義される。この特性を活かすと、従来のオペレーティングシステムに比較して、システムを柔軟なものとすることができる。

【０００４】

すなわち、オブジェクト指向オペレーティングシステムでは、例えば、オペレーティングシステムのサービスを提供するオブジェクトを、オペレーティングシステムの実行開始前に適切に組み合わせることで、ユーザの必要な機能に合わせたシステムを容易に構築することができる。また、システムの最適化やアップデート等のためになされる機能の追加や削除を、システムを停止することなく動的に行うようなことも可能になる。このように、オブジェクト指向オペレーティン

グシステムは、システムコンフィグレーションの柔軟性や、システムの動的変更の容易性などの点で非常に優れている。

【０００５】

ところで、オブジェクト指向オペレーティングシステムのサービスを提供するオブジェクトは、システムオブジェクトと呼ばれる。システムオブジェクト同士は、互いに並行に動作する。そして、メッセージ通信機構を用いて互いに通信を行い、メッセージの交換を行ったり、互いの動作の同期をとったりする。このようなシステムオブジェクトの動作は、システムの部品としてのオブジェクトの独立性を高め、システムコンフィギュレーションの柔軟性や、システムの動的変更の容易性を向上させる。換言すれば、オブジェクト指向オペレーティングシステムのサービス提供部分を、追加や削除が可能なシステムオブジェクトで実現することで、システムコンフィギュレーションの柔軟性や、システムの動的変更の容易性を向上させることができる。

【０００６】

【発明が解決しようとする課題】

以上のように、オブジェクト指向オペレーティングシステムは優れた特徴を有している。しかしながら、オブジェクト指向オペレーティングシステムは、オブジェクト間のメッセージ通信が頻繁になると、通信コストの増大によって、システム全体の実行性能が劣化してしまうという問題をもっている。

【０００７】

したがって、オブジェクト指向オペレーティングシステムのシステム設計者は、システムコンフィギュレーションの柔軟性やシステムの動的変更の容易性など、オブジェクト指向オペレーティングシステムの優れた特徴を取り入れるだけでなく、そのような優れた特徴と実際の実行性能とのバランスを十分に考慮して、システムオブジェクトを設計する必要がある。

【０００８】

本発明は、以上のような従来の実情に鑑みて提案されたものであり、オブジェクト指向オペレーティングシステムの優れた特徴を保ちつつ、システム全体の実行性能を向上することを目的としており、具体的には、そのような機能を実現す

るデータ処理装置及びデータ処理方法、並びにそのような機能を実現するデータ処理プログラムを提供するプログラム提供媒体を提供することを目的としている。

【０００９】

【課題を解決するための手段】

本発明に係るデータ処理装置は、メッセージ通信を行う複数のオブジェクトから構成されるオブジェクト指向オペレーティングシステムを実行するデータ処理装置である。そして、所定のオブジェクトの追加を要求する複合化要求メッセージを受信したオブジェクトが、当該所定のオブジェクトを構成オブジェクトとして参照するためのテーブルデータ構造を作成し、当該構成オブジェクトのデータによって初期化することによって複合オブジェクトを構成する手段を備える。また、少なくとも一つの当該構成オブジェクトのデータ構造を作成し、当該テーブルデータ構造に登録し、当該構成オブジェクトが具備する少なくとも一つのメッセージ処理機能と当該メッセージ処理機能を要求するためのメッセージインターフェースとの関係を、当該構成オブジェクトデータ構造に登録する手段を備える。

【００１０】

なお、上記データ処理装置において、上記複合オブジェクトは、上記テーブルデータ構造を初期化するために、上記構成オブジェクトの名前と、当該構成オブジェクトの有する上記メッセージインターフェースの数と、当該構成オブジェクトを初期化する処理機能とを記載した、所定の設定のデータ構造を読み込む。

【００１１】

また、上記データ処理装置において、上記複合オブジェクトは、独自の実行スレッドを持つことにより、上記構成オブジェクトに対して発行されたメッセージの処理を、当該実行スレッドにおいて実行する。

【００１２】

また、上記データ処理装置において、上記複合オブジェクトは、所定の構成オブジェクトの追加要求を受理することにより、上記テーブルデータ構造に、当該所定の構成オブジェクトのデータ構造を追加登録する。

【００１３】

また、上記データ処理装置は、上記複合オブジェクトに所定のオブジェクトを追加する際、当該複合オブジェクトを構成する全ての構成オブジェクトとの間に、実行逐次性があることを検査する手段と、当該実行逐次性が確認された後、上記所定のオブジェクトを追加する手段とを備える。

【００１４】

また、上記データ処理装置において、上記実行逐次性は、上記所定のオブジェクトにメッセージが送信された時点で、上記複合オブジェクトを構成する全ての構成オブジェクトが、当該所定のオブジェクトと並行に動作する必要がないことを検査する手段と、当該所定のオブジェクトが、当該複合オブジェクトを構成するいずれかの構成オブジェクトに対しメッセージを送信する場合に、当該構成オブジェクトが既に他のメッセージを処理中であることはないことを検査する手段と、当該複合オブジェクトを構成するいずれかの構成オブジェクトから、当該所定のオブジェクトがメッセージを受け取る場合に、当該所定のオブジェクトが既に他のメッセージを処理中であることはないことを検査する手段とによって検査される。

【００１５】

また、上記データ処理装置において、上記構成オブジェクトは、その他全ての上記構成オブジェクトとの間で、実行逐次性があることを検査する。

【００１６】

また、上記データ処理装置において、上記構成オブジェクトは、所定の構成オブジェクトの分離要求を受理することにより、上記テーブルデータ構造から、当該所定の構成オブジェクトのデータ構造を登録抹消する。

【００１７】

また、上記データ処理装置において、上記複合オブジェクトは、当該複合オブジェクトに送達されたメッセージの送付元が、非構成オブジェクトからのものであるか、当該構成オブジェクトからのものであるかを検査する手段と、当該メッセージの送付先が、当該非構成オブジェクトに対するものであるか、当該構成オブジェクトに対するものであるかを検査する手段とを具備する。

【００１８】

また、上記データ処理装置において、上記複合オブジェクトは、上記メッセージの送付元が上記構成オブジェクトであり、当該メッセージの送付先が当該構成オブジェクトに対するものである場合、実行スレッドの切り替えをせずに、メッセージで要求された処理を実行する。

【００１９】

また、本発明に係るデータ処理方法は、メッセージ通信を行う複数のオブジェクトから構成されるオブジェクト指向オペレーティングシステムのデータ処理方法である。そして、所定のオブジェクトの追加を要求する複合化要求メッセージを受信したオブジェクトが、当該所定のオブジェクトを構成オブジェクトとして参照するためのテーブルデータ構造を作成し、当該構成オブジェクトのデータによって初期化することによって複合オブジェクトを構成するステップと、少なくとも一つの当該構成オブジェクトのデータ構造を作成し、当該テーブルデータ構造に登録し、当該構成オブジェクトが具備する少なくとも一つのメッセージ処理機能と当該メッセージ処理機能を要求するためのメッセージインターフェースとの関係を、当該構成オブジェクトデータ構造に登録するステップとを具備する。

【００２０】

なお、上記データ処理方法において、上記複合オブジェクトは、上記テーブルデータ構造を初期化するために、上記構成オブジェクトの名前と、当該構成オブジェクトの有する上記メッセージインターフェースの数と、当該構成オブジェクトを初期化する処理機能とを記載した、所定の設定のデータ構造を読み込む。

【００２１】

また、上記データ処理方法において、上記複合オブジェクトは、独自の実行スレッドを持つことにより、上記構成オブジェクトに対して発行されたメッセージの処理を、当該複合オブジェクトの実行スレッドにおいて実行する。

【００２２】

また、上記データ処理方法において、上記複合オブジェクトは、所定の構成オブジェクトの追加要求を受理することにより、上記テーブルデータ構造に、当該所定の構成オブジェクトのデータ構造を追加登録する。

【００２３】

また、上記データ処理方法において、上記複合オブジェクトは、所定のオブジェクトを追加する際、当該複合オブジェクトを構成する全ての構成オブジェクトとの間に、実行逐次性があることを検査する検査ステップと、当該実行逐次性が確認された後、上記所定のオブジェクトを追加する追加ステップとを実行する。

【００２４】

また、上記データ処理方法において、上記実行逐次性は、上記所定のオブジェクトにメッセージが送信された時点で、上記複合オブジェクトを構成する全ての構成オブジェクトが、当該所定のオブジェクトと並行に動作する必要がないことを検査するステップと、当該所定のオブジェクトが、当該複合オブジェクトを構成するいずれかの構成オブジェクトに対しメッセージを送信する場合に、当該構成オブジェクトが既に他のメッセージを処理中であることはないことを検査するステップと、当該複合オブジェクトを構成するいずれかの構成オブジェクトから、当該所定のオブジェクトがメッセージを受け取る場合に、当該所定のオブジェクトが既に他のメッセージを処理中であることはないことを検査するステップとによって検査される。

【００２５】

また、上記データ処理方法において、上記構成オブジェクトは、その他全ての上記構成オブジェクトとの間で、実行逐次性があることを検査する。

【００２６】

また、上記データ処理方法において、上記構成オブジェクトは、所定の構成オブジェクトの分離要求を受理することにより、上記テーブルデータ構造から、当該所定の構成オブジェクトのデータ構造を登録抹消する。

【００２７】

また、上記データ処理方法において、上記複合オブジェクトは、当該複合オブジェクトに送達されたメッセージの送付元が、非構成オブジェクトからのものであるか、当該構成オブジェクトからのものであるかを検査するステップと、当該メッセージの送付先が、当該非構成オブジェクトに対するものであるか、当該構成オブジェクトに対するものであるかを検査するステップとを具備する。

【００２８】

また、上記データ処理方法において、上記複合オブジェクトは、上記メッセージの送付元が上記構成オブジェクトであり、当該メッセージの送付先が当該構成オブジェクトに対するものである場合、実行スレッドの切り替えをせずに、メッセージで要求された処理を実行する。

【００２９】

また、本発明に係るプログラム提供媒体は、メッセージ通信を行う複数のオブジェクトから構成されるオブジェクト指向オペレーティングシステムのデータ処理プログラムを提供するものであり、所定のオブジェクトの追加を要求する複合化要求を受理したオブジェクトが、当該所定のオブジェクトを構成オブジェクトとして参照するためのテーブルデータ構造を作成し、当該構成オブジェクトのデータによって初期化することによって複合オブジェクトを構成するステップと、少なくとも一つの当該構成オブジェクトのデータ構造を作成し、当該テーブルデータ構造に登録し、当該構成オブジェクトが具備する少なくとも一つのメッセージ処理機能と当該メッセージ処理機能を要求するためのメッセージインターフェースとの関係を、当該構成オブジェクトデータ構造に登録するステップとを具備するデータ処理プログラムを提供する。

【００３０】

また、本発明に係るデータ処理装置は、オブジェクト指向オペレーティングシステムを実行するデータ処理装置であって、オブジェクト間でメッセージ通信を行うオブジェクトを、１つ以上の構成オブジェクトから構成される複合オブジェクトと、複合オブジェクト以外のオブジェクトである標準オブジェクトとのいずれかにより構成するオブジェクト構成手段を備える。また、上記オブジェクト構成手段により構成された標準オブジェクト及び構成オブジェクトを任意のオブジェクトから参照できるように、各標準オブジェクト及び各構成オブジェクトに識別子を付す識別子設定手段を備える。また、上記オブジェクト構成手段により構成されたオブジェクトのうち、複合オブジェクトについては、１つの複合オブジェクトを１つの実行スレッドによって実行し、その実行スレッドを複合オブジェクトを構成する各構成オブジェクトによって共有させる実行スレッド制御手段を

備える。

【００３１】

なお、上記データ処理装置は更に、追加オブジェクト情報読み込み手段と、追加オブジェクト特定手段と、オブジェクト追加手段とを備えていても良い。ここで、追加オブジェクト情報読み込み手段は、所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加することを要求するメッセージが入力されたときに、少なくとも、上記所定のオブジェクトを特定する追加対象オブジェクト名と、当該所定のオブジェクトを構成オブジェクトとして他のオブジェクトに追加するのに必要な初期化手続きが記述されたメソッドを特定する初期化メソッド情報とを読み込む。追加オブジェクト特定手段は、上記追加オブジェクト情報読み込み手段により読み込まれた上記追加対象オブジェクト名により、他のオブジェクトに構成オブジェクトとして追加するオブジェクトを特定する。オブジェクト追加手段は、上記追加オブジェクト特定手段により特定されたオブジェクトを、上記追加オブジェクト情報読み込み手段により読み込まれた上記初期化メソッド情報によって特定されるメソッドを実行することにより、構成オブジェクトとして他のオブジェクトに追加する。

【００３２】

また、上記データ処理装置は更に、記述子作成手段と、メソッド情報格納手段とを備えていてもよい。ここで、記述子作成手段は、所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加する際に、構成オブジェクトとして追加されるオブジェクトに関する情報が格納される記述子を上記識別子と対応づけて作成する。メソッド情報格納手段は、上記記述子作成手段によって作成された記述子に、少なくとも、構成オブジェクトとして追加されるオブジェクトが具備するメソッドを呼び出すための情報を格納する。

【００３３】

また、上記データ処理装置は更に、削除オブジェクト情報読み込み手段と、削除オブジェクト特定手段と、オブジェクト削除手段と、記述子削除手段とを備えていてもよい。ここで、削除オブジェクト情報読み込み手段は、所定の構成オブジェクトを複合オブジェクトから削除することを要求するメッセージが入力され

たときに、少なくとも、削除対象の構成オブジェクトを特定する削除対象オブジェクト名を読み込む。削除オブジェクト特定手段は、上記削除オブジェクト情報読み込み手段により読み込んだ削除対象オブジェクト名により、複合オブジェクトから削除する構成オブジェクトを特定する。オブジェクト削除手段は、上記削除オブジェクト特定手段により特定された構成オブジェクトを複合オブジェクトから削除する。記述子削除手段は、上記削除オブジェクト特定手段により特定された構成オブジェクトに対応した識別子から、当該構成オブジェクトに対応した記述子を特定し、当該記述子を削除する。

【００３４】

また、上記データ処理装置において、上記オブジェクト構成手段は、複合オブジェクトを複数の構成オブジェクトから構成する際に、ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージが送信された時点で、それらの２つの構成オブジェクトが並行動作する必要がないという条件と、ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージを送信するときに、メッセージを受け取る側の構成オブジェクトが他のメッセージを処理中であることはないという条件と、を満たすように複合オブジェクトを構成する。

【００３５】

また、上記データ処理装置において、あるオブジェクトから他のオブジェクトにメッセージを送るときに、メッセージ送信側オブジェクト及びメッセージ受信側オブジェクトが構成オブジェクトであり、それらの構成オブジェクトが同じ複合オブジェクトに含まれている場合、上記実行スレッド制御手段は、実行スレッドの切り替えを行わずに、メッセージ送信側オブジェクトが使用していた実行スレッドと同じ実行スレッドを用いて、メッセージ送信側オブジェクトから送られたメッセージで要求された処理を、メッセージ受信側オブジェクトにより実行させる。

【００３６】

また、上記データ処理装置は更に、オブジェクト間でのメッセージ通信に使用

されるアプリケーションプログラムインターフェースとして、メッセージ通信を行うオブジェクトが標準オブジェクトであるか構成オブジェクトであるかに関わらず共通に使用可能なアプリケーションプログラムインターフェースを備えていてもよい。

【００３７】

また、本発明に係るデータ処理方法は、オブジェクト指向オペレーティングシステムによって実行されるデータ処理方法において、オブジェクト間でメッセージ通信を行うオブジェクトを、１つ以上の構成オブジェクトから構成される複合オブジェクトと、複合オブジェクト以外のオブジェクトである標準オブジェクトとのいずれかにより構成する。また、任意のオブジェクトから標準オブジェクト及び構成オブジェクトを参照できるように、各標準オブジェクト及び各構成オブジェクトに識別子を付す。また、複合オブジェクトについては、１つの複合オブジェクトを１つの実行スレッドによって実行し、その実行スレッドを複合オブジェクトを構成する各構成オブジェクトによって共有させる。

【００３８】

なお、上記データ処理方法において、所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加することを要求するメッセージが入力されたときは、少なくとも、上記所定のオブジェクトを特定する追加対象オブジェクト名と、当該所定のオブジェクトを構成オブジェクトとして他のオブジェクトに追加するのに必要な初期化手続きが記述されたメソッドを特定する初期化メソッド情報とを読み込む。そして、上記追加対象オブジェクト名により、他のオブジェクトに構成オブジェクトとして追加するオブジェクトを特定し、上記追加対象オブジェクト名により特定されたオブジェクトを、上記初期化メソッド情報によって特定されるメソッドを実行することにより、構成オブジェクトとして他のオブジェクトに追加する。

【００３９】

また、上記データ処理方法において、所定のオブジェクトを他のオブジェクトに構成オブジェクトとして追加する際は、構成オブジェクトとして追加されるオブジェクトに関する情報が格納される記述子を上記識別子と対応づけて作成し、

上記記述子に少なくとも、構成オブジェクトとして追加されるオブジェクトが具備するメソッドを呼び出すための情報を格納する。

【００４０】

また、上記データ処理方法において、所定の構成オブジェクトを複合オブジェクトから削除することを要求するメッセージが入力されたときは、少なくとも、削除対象の構成オブジェクトを特定する削除対象オブジェクト名を読み込み、上記削除対象オブジェクト名により、複合オブジェクトから削除する構成オブジェクトを特定する。そして、上記削除対象オブジェクト名により特定された構成オブジェクトを複合オブジェクトから削除する。また、上記削除対象オブジェクト名により特定された構成オブジェクトに対応した識別子から、当該構成オブジェクトに対応した記述子を特定し、当該記述子も削除する。

【００４１】

また、上記データ処理方法において、複合オブジェクトを複数の構成オブジェクトから構成する際は、ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージが送信された時点で、それらの２つの構成オブジェクトが並行動作する必要がないという条件と、ある構成オブジェクトから、当該構成オブジェクトを含む複合オブジェクトを構成する他の構成オブジェクトにメッセージを送信するときに、メッセージを受け取る側の構成オブジェクトが他のメッセージを処理中であることはないという条件と、を満たすように複合オブジェクトを構成する。

【００４２】

また、上記データ処理方法において、あるオブジェクトから他のオブジェクトにメッセージを送るときに、メッセージ送信側オブジェクト及びメッセージ受信側オブジェクトが構成オブジェクトであり、それらの構成オブジェクトが同じ複合オブジェクトに含まれている場合は、実行スレッドの切り替えを行わずに、メッセージ送信側オブジェクトが使用していた実行スレッドと同じ実行スレッドを用いて、メッセージ送信側オブジェクトから送られたメッセージで要求された処理を、メッセージ受信側オブジェクトにより実行する。

【００４３】

また、上記データ処理方法において、オブジェクト間でメッセージ通信を行う際には、メッセージ通信を行うオブジェクトが標準オブジェクトであるか構成オブジェクトであるかに関わらず共通に使用可能なアプリケーションプログラムインターフェースを用いる。

【００４４】

また、本発明に係るプログラム提供媒体は、オブジェクト指向オペレーティングシステムのデータ処理プログラムを提供するプログラム提供媒体である。そして、上記オブジェクト指向オペレーティングシステムのデータ処理プログラムは、オブジェクト間でメッセージ通信を行うオブジェクトを、１つ以上の構成オブジェクトから構成される複合オブジェクトと、複合オブジェクト以外のオブジェクトである標準オブジェクトとのいずれかにより構成する。また、任意のオブジェクトから標準オブジェクト及び構成オブジェクトを参照できるように、各標準オブジェクト及び各構成オブジェクトに識別子を付す。また、複合オブジェクトについては、１つの複合オブジェクトを１つの実行スレッドによって実行し、その実行スレッドを複合オブジェクトを構成する各構成オブジェクトによって共有させる。

【００４５】

【発明の実施の形態】

以下、本発明の実施の形態について、図面を参照しながら詳細に説明する。

【００４６】

１．ハードウェア環境

まず、本発明が適用されるハードウェア構成の一例について、図１を参照して説明する。なお、ここでは、本発明の実施の形態の一例として、テレビジョン受信装置に本発明を適用した例を挙げるが、当然の事ながら、本発明は、その他のデータ処理装置にも適用可能である。すなわち、本発明は、オペレーティングシステムが搭載されるデータ処理装置に広く適用可能であり、例えば、テレビジョン受信装置以外のオーディオ・ビジュアル機器（いわゆるＡＶ機器）や、各種の事務機器や、一般のコンピュータ装置等にも適応可能である。

【００４７】

本発明が適用されたデータ処理装置である図１に示すテレビジョン受信装置は、アンテナ又はケーブル等によって放送局からの信号を受信し、当該信号に基づいて、ブラウン管又は液晶等の画像表示装置に映像を表示すると共にスピーカから音声を出力する。

【００４８】

このテレビジョン受信装置は、通常のテレビ機能を備えているだけでなく、外部からプログラムやデータを受けとることが可能となっており、図１に示すように、バス／ＩＯブリッジ１を介してバス２に接続されたテレビ機能部３と、バス／メモリブリッジ４を介してバス２に接続されたプロセッサ５と、バス／メモリブリッジ４を介してプロセッサ５に接続されたＲＯＭ（Read Only Memory）６及びＲＡＭ（Random Access Memory）７と、バス２に接続された操作パネル８、外部記憶装置９及び通信装置１０とを備えている。

【００４９】

テレビ機能部３は、アンテナ又はケーブル等によって受信した信号に基づいて、映像や音声を再生する機能を備えている。このテレビ機能部３は、バス／ＩＯブリッジ１を介してバス２に接続されており、これにより、他の部分との信号のやり取りが可能となっている。

【００５０】

プロセッサ５は、このテレビジョン受信装置の各部の制御を行うものであり、バス／メモリブリッジ４を介してバス２に接続されている。また、プロセッサ５には、バス／メモリブリッジ４を介してＲＯＭ６及びＲＡＭ７が接続されている。ＲＯＭ６は、プロセッサ５による制御を行うためのオペレーティングシステムやアプリケーションプログラムを記憶している。ＲＡＭ７は、プロセッサ５のワークエリアとして使われる。すなわち、プロセッサ５は、ＲＯＭ６に記憶されているオペレーティングシステムやアプリケーションプログラムを、ＲＡＭ７をワークエリアとして使用して実行することにより、このテレビジョン受信装置を構成する各部を制御する。

【００５１】

操作パネル８は、ユーザからの操作入力を受け付けるための入力装置であり、この操作パネル８から、例えば、テレビのチャンネルやボリューム等の切り換えを指示する信号が入力される。この操作パネル８は、具体的には、各種信号を入力するための複数のボタンを備えた入力装置や、いわゆるマウスと称されるようなポインティングデバイス等からなる。この操作パネル８によって入力された信号は、バス２及びバス／メモリブリッジ４を介してプロセッサ５に入力される。そして、プロセッサ５は、操作パネル８によって入力された信号に基づいて、所定の演算処理を行って各部を制御する。

【００５２】

外部記憶装置９は、例えばハードディスク装置からなり、画像データ、制御データ、又は外部から通信装置１０を介してダウンロードされたプログラム等を記録するのに使われる。また、通信装置１０は、外部との間でデータ通信を行うための入出力部であり、例えばモデムやターミナルアダプター等からなる。

【００５３】

このテレビジョン受信装置は、テレビ機能部３によって提供される通常のテレビ機能を備えているだけでなく、通信装置１０を介して、外部からプログラムやデータを受け取ることが可能となっている。例えば、このテレビジョン受信装置では、オペレーティングシステムのバージョンアップを行う際に、外部のネットワークから通信装置１０を介して新規ソフトウェアモジュールを受け取り、これにより、オペレーティングシステムのバージョンアップを行うことが可能となっている。

【００５４】

また、このテレビジョン受信装置では、プロセッサ５によって、ＲＯＭ６に記憶されているオペレーティングシステムを実行し、このオペレーティングシステム上で、ＲＯＭ６や外部記憶装置９に記憶されているアプリケーションプログラムを実行することにより、各部の制御を行う。すなわち、このテレビジョン受信装置は、オペレーティングシステムのデータ処理プログラムを提供するプログラム提供媒体として、ＲＯＭ６を備えている。なお、オペレーティングシステムは

、ＲＡＭ７や外部記憶装置９に記録しておくようにしてもよい。特に、オペレーティングシステムの書き換えを行えるようにしたい場合には、ＲＡＭ７や外部記憶装置９に記録しておいたほうが好ましい。

【００５５】

本例におけるオペレーティングシステムは、純オブジェクト指向型オペレーティングシステムである。そして、このオペレーティングシステム上で、例えば、テレビ機能部３に動画像を表示するためのアプリケーションプログラムや、操作パネル８を制御するためのグラフィカル・ユーザ・インターフェース（ＧＵＩ）を実現するアプリケーションプログラムが実行される。

【００５６】

２．ソフトウェア環境

つぎに、図１に示したようなテレビ受信装置装置で使用される、本発明を適用したオペレーティングシステムについて詳細に説明する。

【００５７】

２－１　オペレーティングシステムの構成

このオペレーティングシステムは、オペレーティングシステムの基本部分であるメタコア（MetaCore）と、その他のオブジェクト群とから構成される。ここで、メタコアは、オブジェクトとしては定義できない部分であり、オブジェクト間の実行制御の切り替えを実行する処理部、すなわち実行スレッドの切り替えを実行するスレッド切替処理部である。

【００５８】

また、メタコアは、ユーザプログラムにより変更が困難な部分であり、一方、その他のオブジェクト群は、ユーザプログラムにより変更が容易な部分である。なお、ここでの「変更」とは、当該変更をオペレーティングシステムに反映させるためには、全てのアプリケーションプログラムを停止して、オペレーティングシステムを再起動させる必要があるような、オペレーティングシステムの変更のことである。

【００５９】

このオペレーティングシステムは、純オブジェクト指向オペレーティングシス

テムであり、オペレーティングシステム上で実行されるアプリケーションプログラムを構成するオブジェクトと、オペレーティングシステムを構成するオブジェクトとが、同様な実行機構を持つオペレーティングシステムである。したがって、オブジェクト間の実行スレッドの切り替えは、サービスの依頼と提供という観点から考えると、次の２種類に集約できる。

【００６０】

（１）サービス依頼者からサービス提供者に実行制御が移る。

【００６１】

（２）サービス提供者からサービス依頼者に実行制御が戻る。

【００６２】

ここで、オペレーティングシステムによって提供されるサービスを依頼する側であるサービス依頼者のオブジェクトのことを、「ベースレベルオブジェクト」と称する。また、オペレーティングシステムによって提供されるサービスを提供する側であるサービス提供者のオブジェクトのことを、「メタオブジェクト」と称する。また、メタオブジェクトにより提供されるサービスのことを、「メタオペレーション」と称する。また、サービス依頼者とサービス提供者との間の関係のことを、「ベース／メタの関係」と呼ぶ。このとき、オペレーティングシステム内の全ての動作は、オブジェクト間のベース／メタの関係で表現することができる。

【００６３】

そして、このオペレーティングシステムでは、上記２種類の実行スレッドの切り替えを行うスレッド切替処理部が、メタコアとして提供される。すなわち、このオペレーティングシステムでは、オペレーティングシステムを構築するための必要最低限の機構が、メタコアによって提供される。

【００６４】

すなわち、このオペレーティングシステムは、実行スレッドに関する情報の書き換えを伴う実行スレッドの切り替えの処理を担うスレッド切替処理部として、メタコアを備えている。そして、このメタコアの内容を変更することなく、当該オペレーティングシステムを構成するオブジェクトの内容変更が可能とされてい

る。

【００６５】

このオペレーティングシステムでは、ユーザが変更困難な部分は最小化され、様々なハードウェア環境やユーザの要求に対して、容易に対応が可能な、柔軟性が高いシステムを実現できる。すなわち、例えば、オブジェクト間でメッセージをやり取りするオブジェクト間通信を担う部分を交換したり更新したりするようなことが、オペレーティングシステムを再起動させること無く行うことができる。したがって、オブジェクト間通信を担う部分のバグの修正や、新デバイスの追加に対する新機能の追加といったことを、非常に容易に行うことが可能である。また、プロセッサのアーキテクチャに依存する部分を、メタコアに集中させることにより、他のシステム構成要素の移植性を高めることもできる。

【００６６】

図２に、このオペレーティングシステムを構成するオブジェクト群の例を示す。図２の例において、オペレーティングシステムは、システムオブジェクトとして、「オペレーティングシステム初期化手続き」、「オブジェクトマネージャ」、「ダウンローダ」、「レジストリ」、「割り込みベクタ」、「ＯＩＤマネージャ」、「スレッドマネージャ」、「メモリマナージャ」、「スケジューリング機構」、「スケジューリングポリシ」、「メッセージハンドラ」、「メモリスイッチャ」、「タイマ」を備えており、これらがメタコア上で動作するようになされている。

【００６７】

なお、図２は、このオペレーティングシステムの典型的な構成例において存在するオブジェクト群を示しており、これらのオブジェクト群は変更される可能性がある。すなわち、このオペレーティングシステムは、ユーザによってシステム構成を柔軟に変更することが可能となっており、当該オペレーティングシステムを構成するオブジェクト群は容易に変更可能とされている。ただし、図２に示したオブジェクト群のうち、オブジェクト「オペレーティングシステム初期化手続き」は必須であり、このオブジェクト「オペレーティングシステム初期化手続き」だけは、オペレーティングシステムを構成するオブジェクト群に必ず含まれる

【００６８】

図２に示したオブジェクト群のうち、オブジェクト「オペレーティングシステム初期化手続き」は、オペレーティングシステムの起動時に最初に実行される初期化手続きを実行する。オブジェクト「オブジェクトマネージャ」は、オブジェクトの生成や削除を管理する。オブジェクト「ダウンローダ」は、ネットワーク、二次記憶装置又は着脱可能メディア等から、オブジェクトをダウンロード又はアンロードするときの手続きを実行する。

【００６９】

オブジェクト「レジストリ」は、各オブジェクトを識別するための識別子（Object identifier：以下、「ＯＩＤ」と称する。）と、各オブジェクトの名称とのマッピングを行う。オブジェクト「割り込みベクタ」は、割り込み発生時に割り込みの種類を判別し、適当な手続きを呼び出す。オブジェクト「ＯＩＤマネージャ」は、ＯＩＤの生成や削除を管理する。オブジェクト「スレッドマネージャ」は、オブジェクトの実行スレッドの生成や削除を管理する。オブジェクト「メモリマナージャ」は、記憶領域の割り当てや解放の手続きを行う。

【００７０】

オブジェクト「スケジューリング機構」は、オブジェクトのスケジューリングを行うときに、実行スレッドの切り替えのタイミングの管理を行い、オブジェクト「スケジューリングポリシ」は、スケジューリングキューの動作を含むスケジューリングの方策の管理を行う。すなわち、オブジェクト「スケジューリング機構」は、スケジューリングの低レベル(すなわち、よりハードウェアに近いレベル)の手続きを扱い、オブジェクト「スケジューリングポリシ」は、スケジューリングの高レベル(すなわち、よりアプリケーションに近いレベル)の手続きを扱う。

【００７１】

オブジェクト「メッセージハンドラ」は、アプリケーションオブジェクト間のメッセージ通信機構を管理する。具体的には、オブジェクト「メッセージハンドラ」は、アプリケーションオブジェクト間でメッセージ通信を行うときに、メッ

セージキューの管理や、メッセージ受信者の確認処理などを行う。

【００７２】

オブジェクト「基本メッセージハンドラ」は、システムオブジェクト間のメッセージ通信機構を管理する。具体的には、オブジェクト「基本メッセージハンドラ」は、システムオブジェクト間でメッセージ通信を行うときに、メッセージキューの管理や、メッセージ受信者の確認処理などを行う。

【００７３】

オブジェクト「メモリスイッチャ」は、ＭＭＵ(Memory Management Unit:メモリ管理機構)を用いて、実行スレッド切り替えのタイミングに合わせて、アクティブなメモリ空間を切り替える。オブジェクト「タイマ」は、ハードウェアタイマの割り込みを管理し、定期的処理を管理する。

【００７４】

２－２　オブジェクトの実行遷移

図３に、オペレーティングシステム上で動作する２つのアプリケーションオブジェクト(アプリケーションオブジェクト１とアプリケーションオブジェクト２）の間でメッセージ通信が行われるときの、システムオブジェクトの実行の遷移を示す。

【００７５】

アプリケーションオブジェクト１でメッセージ送信要求が起こると、先ず、システムオブジェクトの一つであるオブジェクト「メッセージハンドラ」への実行スレッドの切り替えが起こる。ここで、アプリケーションオブジェクトとシステムオブジェクトの間ではメッセージ通信は起こらず、メタコアによる実行スレッドの切り替えだけが起こる。これは、アプリケーションオブジェクトとシステムオブジェクトの動作レベルを切り替えることで、オペレーティングシステムとアプリケーションとの間に境界を作り、システムの安全性を高めるためである。

【００７６】

次に、オブジェクト「メッセージハンドラ」は、メッセージ受信者であるアプリケーションオブジェクト２に対するメッセージキューの処理を行う。そして、処理中のメッセージがキューに貯まっていなければ、オブジェクト「メモリスイ

ッチャ」を起動する。また、処理中の他のメッセージがある場合には、アプリケーションオブジェクト２に対するメッセージをキューに格納し、当該メッセージを処理する順番が来たら、オブジェクト「メモリスイッチャ」を起動する。

【００７７】

次に、オブジェクト「メモリスイッチャ」は、アクティブなメモリ空間をアプリケーションオブジェクト１のメモリ空間から、アプリケーションオブジェクト２のメモリ空間に切り替える。

【００７８】

次に、オブジェクト「メッセージハンドラ」は、アプリケーションオブジェクト２の実行スレッドの情報を、オブジェクト「スケジューリングポリシ」に受け渡す。

【００７９】

次に、オブジェクト「スケジューリングポリシ」及びオブジェクト「スケジューリング機構」により、スケジューリングキューの処理を行う。このとき、キューにはアプリケーションオブジェクト１の実行スレッドが存在するはずなので、その後ろにアプリケーションオブジェクト２の実行スレッドを格納する。

【００８０】

その後、実行の制御はオブジェクト「メッセージハンドラ」まで戻り、続いてアプリケーションオブジェクト１への実行スレッドの切り替えが起こり、アプリケーションオブジェクト１の実行が再開される。

【００８１】

やがて、オブジェクト「タイマ」が、ハードウェア割込み発生により起動され、オブジェクト「タイマ」によりオブジェクト「スケジューリング機構」が呼び出される。そして、アプリケーションオブジェクト２へ実行スレッドの切り替えが起こり、メッセージ受信者であるアプリケーションオブジェクト２も動作を開始する。

【００８２】

以後は、タイムスライススケジューリングの元で、アプリケーションオブジェクト１とアプリケーションオブジェクト２とが並行に動作する。

【００８３】

なお、以上の例は、アプリケーションオブジェクト間でメッセージ通信を行うときのオブジェクトの実行遷移であったが、システムオブジェクト間でメッセージ通信を行うときも、ほぼ同様にオブジェクトの実行遷移がなされる。

【００８４】

ただし、システムオブジェクト間でのメッセージ通信を処理するオブジェクトは、オブジェクト「メッセージハンドラ」とは異なるオブジェクトであるオブジェクト「基本メッセージハンドラ」である。また、システムオブジェクト同士は、通常は同一のメモリ空間に配置されているので、システムオブジェクト間でメッセージ通信を行うとき、オブジェクト「メモリスイッチャ」は動作しない。

【００８５】

システムオブジェクト間でメッセージ通信を行うときの実行遷移の例として、図４に、オブジェクト「基本メッセージハンドラ」とオブジェクト「スケジューリングポリシ」との間でメッセージ通信を行うときの実行遷移を示す。

【００８６】

図４に示すように、システムオブジェクト間でメッセージ通信を行うときも、その実行遷移は、オブジェクト「メッセージハンドラ」ではなくオブジェクト「基本メッセージハンドラ」が動作することと、オブジェクト「メモリスイッチャ」が動作しないこと以外については、図３に示したアプリケーションオブジェクト間でメッセージ通信を行う場合の実行遷移と同様である。

【００８７】

なお、図４の例において、メッセージ受信者であるオブジェクト「スケジューリングポリシ」のスケジューリングは、オブジェクト「スケジューリングポリシ」自身で行っている。すなわち、オブジェクト「スケジューリングポリシ」についても、そのスケジューリングを行うオブジェクトは、オブジェクト「スケジューリングポリシ」である。オブジェクト「スケジューリングポリシ」は、このようなモデルに矛盾しないように、自分自身をアクティブにするときには、他の場合とは区別して例外的な処理を行う。

【００８８】

２－３　メッセージ通信機構

以下、メッセージ通信を行う機構について、具体的なアプリケーションプログラムインタフェース(Application Program Interface：以下、ＡＰＩと称する。)を例に挙げて詳細に説明する。

【００８９】

なお、以下の説明では、ＡＰＩをＯＭＧ　ＩＤＬ（Object Management Group Interface Definition Language）での記述方法に準じた形で示し、その後、そのＡＰＩについて説明する。

【００９０】

２－３－１　メッセージ通信に用いられるＡＰＩ

まず、オブジェクト間のメッセージ通信に用いられるＡＰＩについて、具体的な例を挙げて説明する。なお、以下に挙げるＡＰＩは、アプリーケーションオブジェクト間でメッセージ通信を行う場合と、システムオブジェクト間でメッセージ通信を行う場合のどちらにおいても使用される。また、以下の説明では、メッセージ送信側のオブジェクトを送信オブジェクトと称し、メッセージ受信側のオブジェクトを受信オブジェクトと称する。

【００９１】

また、以下に挙げるＡＰＩにおいて、「sError」はエラー識別値を表す変数型を示している。また、引数型の前に「in」とあるのは入力引数、「out」とあるのは出力引数である。また、変数型「OID」は、ＯＩＤを表す変数型であり、変数型「Selector」は、メソッドに対応したメッセージセレクタを表す変数型である。

【００９２】

また、変数型「FutureID」は、フューチャ構造体の識別子を表す変数型である。フューチャ構造体は、送信オブジェクトと受信オブジェクトの同期をとるために用いられる構造体であり、フューチャ構造体には、受信オブジェクトによる処理の結果として送信オブジェクトに返されるメッセージが格納される。そして、各フューチャ構造体には、個々のフューチャ構造体を識別するための識別子が付

けられる。

【００９３】

sError FindOID (in char* name, out OID* object);

このＡＰＩは、入力引数「name」で与えたオブジェクト名に対応するオブジェクトのＯＩＤを出力引数「object」に代入する。なお、本例におけるオペレーティングシステムでは、オブジェクトの参照にＯＩＤを用いる。そして、オブジェクト名とＯＩＤのマッピングは、システムオブジェクトの一つであるオブジェクト「レジストリ」によって管理される。

【００９４】

sError Send (in OID destination, in Selector method,
in void* arg, out FutureID* future);

このＡＰＩは、送信オブジェクトから、入力引数「destination」で特定される受信オブジェクトに対して、メッセージを送信する。また、入力引数「method」で特定される受信オブジェクトのメソッドを起動し、当該メソッドへの引数として入力引数「arg」を受け渡す。また、メッセージ通信に使用されるフューチャ構造体の識別子を、戻り値として出力引数「future」に格納する。

【００９５】

受信オブジェクトがメッセージを受信した後、送信オブジェクトと受信オブジェクトは並行に動作する。また、送信オブジェクトから受信オブジェクトにメッセージが送信された時に、受信オブジェクトが別のメッセージを処理中の場合は、その処理が終了するまで、送信されたメッセージの処理は開始されず、そのメッセージはメッセージキューに格納される。

【００９６】

sError WaitFor (in FutureID future);

このＡＰＩは、送信オブジェクトと受信オブジェクトの同期をとるためのものであり、入力引数「future」で特定されるフューチャ構造体に、受信オブジェクトによる処理の結果として送信オブジェクトに返されるメッセージが格納されるまで、送信オブジェクトの処理を中断する。すなわち、このＡＰＩが発行された場合、受信オブジェクトから返答があるまで、送信オブジェクトは処理を中断す

る。

【００９７】

```
sError Reply ();
```

このＡＰＩは、受信オブジェクトによる処理の結果として、送信オブジェクトにメッセージを返す。なお、送信オブジェクトに返すメッセージは、このＡＰＩの引数とするのではなく、フューチャ構造体に格納することで送信オブジェクトに受け渡す。したがって、このＡＰＩでは、送信オブジェクトに返すメッセージを代入する引数を用意していない。

【００９８】

```
void Exit ();
```

このＡＰＩは、オブジェクトのメソッド実行を終了する。このＡＰＩが発行された時点でメッセージキューに格納されているメッセージがある場合は、そのメッセージの処理が開始される。

【００９９】

以上のようなＡＰＩを使った場合の実行遷移の一例を図５に示す。

【０１００】

図５は、送信オブジェクト「Sender」から受信オブジェクト「Receiver」へメッセージを送信する場合を示している。具体的には、送信オブジェクト「Sender」のメソッド「Method1()」が、受信オブジェクト「Receiver」のメソッド「Method2(ptr)」の起動を要求するメッセージを送信し、その後、送信オブジェクト「Sender」と受信オブジェクト「Receiver」の同期をとって、受信オブジェクト「Receiver」のメソッド「Method2(ptr)」での処理の結果を送信オブジェクト「Sender」に返す、という処理の実行遷移を示している。

【０１０１】

図５に示した例では、先ず、「FindOID("Receiver",&receiver_oid)」により、"Receiver"という名前のオブジェクトのＯＩＤが、出力引数「receiver_oid」に代入される。

【０１０２】

次に、「Send(receiver_oid,method2_selector,pArgument,&futureID)」によ

り、受信オブジェクト「Receiver」のメソッド「Method2(ptr)」の起動を要求するメッセージが、送信オブジェクト「Sender」から受信オブジェクト「Receiver」へ送信される。ここで、受信オブジェクト「Receiver」のメソッド「Method2(ptr)」は、引数「method2_selector」によって特定される。また、引数「pArgument」は、受信オブジェクト「Receiver」のメソッド「Method2(ptr)」に受け渡される。また、受信オブジェクト「Receiver」による処理の結果として送信オブジェクト「Sender」に返されるメッセージが格納されるフューチャ構造体を識別するための識別子が、戻り値として出力引数「futureID」に格納される。

【０１０３】

そして、「Send(receiver_oid,method2_selector,pArgument,&futureID)」が発行されると、図３や図４に示したような処理がシステムオブジェクトによって行われ、その結果、受信オブジェクト「Receiver」が起動される。この時点から、送信オブジェクト「Sender」と受信オブジェクト「Receiver」は並行動作する。

【０１０４】

その後、送信オブジェクト「Sender」が「WaitFor(futureID)」を発行し、受信オブジェクト「Receiver」からの返答待ちの状態になると、送信オブジェクト「Sender」の実行は、受信オブジェクト「Receiver」が「Reply()」を発行するまで中断される。

【０１０５】

その後、受信オブジェクト「Receiver」が「Reply()」を発行し、送信オブジェクト「Sender」への返答を行うと、送信オブジェクト「Sender」の実行が再開され、送信オブジェクト「Sender」と受信オブジェクト「Receiver」は再び並行動作する。なお、受信オブジェクト「Receiver」から送信オブジェクト「Sender」への返答メッセージの受け渡しは、フューチャ構造体を介して行われる。

【０１０６】

そして、以上のようなメッセージ通信の後、やがて両者とも「Exit()」を発行することで、処理を終了する。

【０１０７】

２－３－２　オブジェクトの外部インターフェース

オブジェクトのメソッドは、エントリテーブルに登録することにより、外部インターフェースとして使用可能となる。オブジェクトの外部インタフェースとは、他のオブジェクトに公開されているメソッドのことであり、換言すれば、メッセージ通信により他のオブジェクトから起動することが可能なメソッドである。例えば、図５の例では、受信オブジェクト「Receiver」のメソッド「Method2(ptr)」は、外部インターフェースの一つである。

【０１０８】

エントリテーブルは、図６に示すように、メッセージセレクタとエントリの組からなるテーブルであり、各オブジェクトに一つずつ保持されている。ここで、エントリは、メソッドへのポインタである。このエントリテーブルは、オブジェクトが起動されるときに、メソッドを検索するために、システムオブジェクトによって参照される。

【０１０９】

以下、このようなエントリーテーブルへのメソッドの登録等の処理を行うＡＰＩについて、具体的な例を挙げて説明する。なお、以下に挙げるＡＰＩは、オブジェクトの初期化時に呼び出される特別なメソッド(以下、「プロローグメソッド」と称する。)において主に用いられるが、動的に外部インタフェースの登録を行う場合に任意のメソッドから呼び出して用いることも可能である。

【０１１０】

なお、以下に挙げるＡＰＩにおいて、「sError」はエラー識別値を表す変数型を示している。また、引数型の前に「in」とあるのは入力引数、「out」とあるのは出力引数である。また、変数型「EntryTable」は、エントリテーブルへのポインタを表す変数型であり、変数型「Selector」は、メソッドに対応したメッセージセレクタを表す変数型である。

【０１１１】

```
sError InitEntryTable (in EntryTable table);
```

このＡＰＩは、エントリテーブルをオブジェクトに登録する。ここで、入力引

数「table」には、オブジェクトに登録するエントリテーブルへのポインタを設定する。このポインタにより特定されるエントリテーブルの内容が、オブジェクトに登録され、当該エントリテーブルに登録されているメソッドが、外部インターフェースとして使用可能となる。

【０１１２】

sError GetEntryTable (out EntryTable table);

このＡＰＩは、オブジェクトに登録されているエントリテーブルの内容を、エントリテーブルの形式で得る。出力引数「table」には、エントリテーブルへのポインタが格納される。

【０１１３】

void EntryTable::SetEntry (in Selector method, in Entry entry);

このＡＰＩは、エントリテーブルにメッセージセレクタやエントリの値を登録する。入力引数「method」には、登録するメッセージセレクタを設定し、入力引数には、登録するエントリを設定する。

【０１１４】

void EntryTable::UnsetEntry (in Selector method);

このＡＰＩは、エントリテーブルから値を削除する。入力引数「method」には、削除するメソッドのメッセージセレクタを設定する。

【０１１５】

下記にコード１として、上述のようなＡＰＩを用いたプロローグメソッドの一例(メソッド名「Prologue」)を示す。

【０１１６】

コード１：外部インターフェースの登録

```
1: void
2: Prologue ()
3: {
4:     EntryTable entryTable (2);
5:
6:     entryTable.SetEntry (selector1, Method1);
```

```
 7:     entryTable.SetEntry (selector2, Method2);
 8:
 9:     InitEntryTable (&entryTable);
10: }
```

上記メソッド「Prologue」は、オブジェクトの初期化時にメソッドを外部インタフェースとして登録するプロローグメソッドである。このメソッド「Prologue」では、４行目において、エントリ数が２であるエントリテーブルを定義し、６，７行目において、エントリテーブルにメッセージセレクタやエントリの値を登録する。そして、メッセージセレクタやエントリの値が登録されたエントリーテーブルを、９行目において、このプロローグメソッドを発行したオブジェクトに登録する。これにより、エントリテーブルに登録されているメソッドが、外部インターフェースとして使用可能となる。

【０１１７】

また、下記にコード２として、上述のようなＡＰＩを用いて外部インターフェースの変更を行うメソッドの一例（メソッド名「ChangeExternalInterface」）を示す。

【０１１８】

コード２：外部インターフェースの変更

```
 1: void
 2: ChangeExternalInterface ()
 3: {
 4:     EntryTable* pEntryTable;
 5:
 6:     GetEntryTable (pEntryTable);
 7:
 8:     entryTable.UnsetEntry (selector1);
 9:     entryTable.SetEntry (selector3, Method1);
10:
11:     InitEntryTable (pEntryTable);
```

12: }

上記メソッド「ChangeExternalInterface」は、上記メソッド「Prologue」で登録したメソッドのうち、メソッド「Method1」について外部インタフェースとしての登録を抹消し、代わりにメッセージセレクタ「selector3」で参照されるエントリ「Method3」を新たに登録する。このメソッド「ChangeExternalInterface」において、４行目で定義している「EntryTable* pEntryTable」には、エントリの交換を行うために用いられるエントリテーブルへのポインタが格納される。そして、このメソッド「ChangeExternalInterface」は、６行目において、エントリテーブルのポインタを獲得し、８行目でメッセージセレクタ「selector1」のエントリを削除し、９行目で新しいエントリを登録する。そして、変更されたエントリテーブルの内容を、１１行目において、オブジェクトに再登録する。

【０１１９】

２－４　動的リンク機構

動的リンク機構について説明する。動的リンク機構は、オブジェクト間で動的にリンク可能な共有のライブラリ（以下、動的共有ライブラリと称する。）を構築するときに用いられたり、後述する複合オブジェクトの動的変更機構の一部として用いられる。

【０１２０】

動的リンク機構は、オブジェクトを複数の「オブレット」と呼ぶソフトウェアモジュールを用いて構成するための機構である。オブレットの構造を、図７のＯＭＴダイアグラム(Object Modeling Technique Diagram)を用いて説明する。

【０１２１】

オブレットは「リンクテーブル(LinkTable)」と任意の数の「アドレススペース(AddressSpace)」から成る。リンクテーブルは「エントリ(Entry)」（このエントリは、後述するテキストセグメント内にあるメソッドへのポインタ)の集合である。アドレススペースは、メモリ保護の単位で、複数の「セグメント(Segment)」から成る。そして、通常は実行バイナリのテキスト(コード）領域及びデータ領域が、それぞれ「テキストセグメント(TextSegment)」及び「データセグメント(DataSegment)」として、オブレット内に定義される。したがって、テキスト

セグメント内にメソッドが格納され、リンクテーブルのエントリは、テキストセグメント内に格納されたメソッドへのポインタとなる。

【０１２２】

全てのオブジェクトはエントリテーブルを持つが、エントリテーブルに格納されたエントリ(メソッドへのポインタ)が指す先は、テキストセグメント中のエントリでもよいし、リンクテーブル中のエントリでもよい。あるオブジェクトのエントリテーブルに格納されたエントリが、リンクテーブル中のエントリを指した場合は、当該リンクテーブルを有するオブレット中のテキストセグメントの内容が、そのオブジェクトのメソッドとして用いられる。

【０１２３】

このような動的リンク機構を利用すると、オブジェクト間で動的にリンク可能な共有のライブラリ、すなわち動的共有ライブラリの構築が可能となる。図８に、オブジェクトＡとオブジェクトＢで、オブレットＣを使った動的共有ライブラリを用いている例を示す。

【０１２４】

オブジェクトＡは３つのエントリを持つエントリテーブルを持っており、それらのうち、メッセージセレクタ「selector1」「selector2」に対応するエントリは、オブジェクトＡのテキストセグメント内のメソッドへのエントリとなっている。また、メッセージセレクタ「selector3」に対応するエントリは、オブレットＣのリンクテーブル内のエントリを指しており、このリンクテーブル内のエントリは、オブレットＣのテキストセグメント内のメソッドへのエントリとなっている。

【０１２５】

一方、オブジェクトＢのエントリテーブルも３つのエントリを持っており、それらのうち、メッセージセレクタ「selector1」に対応するエントリは、オブジェクトＢのテキストセグメント内のメソッドへのエントリとなっている。また、メッセージセレクタ「selector2」「selector3」に対応するエントリは、オブレットＣのリンクテーブル内のエントリを指しており、このリンクテーブル内のエントリは、オブレットＣのテキストセグメント内のメソッドへのエントリとなっ

ている。

【０１２６】

このとき、オブレットＣは、オブジェクトＡ及びオブジェクトＢからリンクされた共有のライブラリとなっている。したがって、例えば、共有ライブラリとなっているオブレットＣを更新することにより、オブジェクトＡとオブジェクトＢの動作を同時に更新することが可能である。また、オブレットＣのリンクテーブルへのリンクは、動的に変更可能である。すなわち、オブレットＣは、オブジェクト間で動的にリンク可能な共有のライブラリ、すなわち動的共有ライブラリとなっている。

【０１２７】

オブレットを動的共有ライブラリとして用いる場合、オブレットのリンクテーブルにどの種類のエントリがあるかを抽出した「オブレット情報」がオブレットの定義時に提供される。オブジェクトの初期化手続きにおいて、オブジェクトが参照しているライブラリ内のエントリ名を、オブレット情報内のエントリ名と比較して一致した場は、そのオブジェクトのエントリテーブルからオブレットのリンクテーブルに対してリンクを張る。

【０１２８】

なお、以上のような動的リンク機構の複合オブジェクトへの利用については後述する。

【０１２９】

２－５　複合オブジェクト

複合オブジェクトについて説明する。複合オブジェクトは、本発明のポイントとなるものであり、複合オブジェクトを導入することにより、オペレーティングシステムの柔軟性を保ったまま、オブジェクト間通信のコストを低減することが可能となる。

【０１３０】

２－５－１　複合オブジェクトの概念

「複合オブジェクト(Composite Object)」は、次の性質を持つオブジェクトである。

【０１３１】

(1)複合オブジェクトは、複数の「構成オブジェクト(Component Object)」から成る。

【０１３２】

(2)構成オブジェクトのプログラミングスタイルは、構成オブジェクト以外のオブジェクト(以下、「標準オブジェクト」と称する。)のプログラミングスタイルと同じである。つまり、構成オブジェクトも、標準オブジェクトと同様に、上述したＡＰＩを用いたメッセージ通信により互いにメッセージをやり取りする実体として定義される。また、プロローグメソッドによる初期化方法についても、構成オブジェクトと標準オブジェクトとで同一のプログラミングスタイルを用いる。

【０１３３】

(3)各構成オブジェクトは、標準オブジェクトと同様にＯＩＤを持ち、構成オブジェクトや標準オブジェクトから参照可能である。

【０１３４】

(4)複合オブジェクト内の構成オブジェクトは、独自の実行スレッドを持たず、複合オブジェクトが持つ実行スレッドを共有する。

【０１３５】

(5)複合オブジェクトは複数の構成オブジェクトから成るが、これらの構成オブジェクト間でのメッセージ通信は、標準オブジェクト間のメッセージ通信よりも高速である。これは、複合オブジェクトを構成している構成オブジェクト間でメッセージ通信を行うときには、実行スレッドの切り替えが起こらずに、関数呼び出しと同様な制御の流れで、オブジェクト間通信が行われるからである。

【０１３６】

(6)２つのオブジェクトが複合オブジェクトの構成オブジェクトになることが可能なのは、両オブジェクトの間に「実行逐次性」がある場合である。ここで、２つのオブジェクトの間に実行逐次性がある場合とは、具体的には、次の２つの条件を満たす場合である。

【０１３７】

(a)一方のオブジェクトから他方のオブジェクトにメッセージが送信された時点で、それらの２つのオブジェクトが並行動作する必要がない。

【０１３８】

(b)一方のオブジェクトから他方のオブジェクトにメッセージを送信するときに、メッセージを受け取る側のオブジェクトが他のメッセージを処理中であることがない。

【０１３９】

(7)複合オブジェクトを標準オブジェクトに変換したり、標準オブジェクトを複合オブジェクトに変換することは可能である。

【０１４０】

(8)構成オブジェクトの外部インタフェースは、当該構成オブジェクトが属する複合オブジェクトの外部インタフェースでもある。

【０１４１】

(9)複合オブジェクトを構成する構成オブジェクトは、複合オブジェクトの初期化時に決定可能であるとともに、その後、動的に追加や削除することも可能である。

【０１４２】

図９に、２つの標準オブジェクトである「オブジェクトＡ」と「オブジェクトＢ」から、複合オブジェクトである「オブジェクトＣ」を構成した例を示す。図９の例において、オブジェクトＡには、メソッド１，２の二つのメソッドが外部インタフェースとして定義されている。また、オブジェクトＢには、メソッド３，４，５の三つのメソッドが外部インタフェースとして定義されている。

【０１４３】

そして、図９中の矢印Ａに示すように、オブジェクトＡとオブジェクトＢを合成し複合化することで、複合オブジェクトＣを構成することができる。このとき、オブジェクトＡ及びオブジェクトＢは、複合オブジェクトＣの構成オブジェクトとなり、メソッド１，２，３，４，５は、複合オブジェクトＣの外部インタフェースとなる。これらの外部インタフェースは、オブジェクトＡ及びオブジェク

トＢが標準オブジェクトであるときと同様である。

【０１４４】

また、オブジェクトＡ及びオブジェクトＢを複合オブジェクトＣの構成オブジェクトとしたとき、それらの構成オブジェクトはそれぞれのＯＩＤを持つ。したがって、それらの構成オブジェクトは、標準オブジェクトと同様に、他の構成オブジェクトや標準オブジェクトから参照可能である。

【０１４５】

また、オブジェクトＡ及びオブジェクトＢを複合オブジェクトＣの構成オブジェクトとしたとき、それらの構成オブジェクトは、複合オブジェクトＣの実行スレッドを共用する。

【０１４６】

また、図９中の矢印Ｂに示すように、オブジェクトＣを分解し複合解除することで、複合オブジェクトＣを構成していた構成オブジェクトを、標準オブジェクトに戻すこともできる。

【０１４７】

ところで、標準オブジェクトを、複合オブジェクトを構成する構成オブジェクトにするには、オブジェクト間の実行逐次性が重要である。

【０１４８】

例えば、３つの標準オブジェクトＡ，Ｂ，Ｃがあり、オブジェクトＡとオブジェクトＢとの間には実行逐次性がなく（換言すれば、オブジェクトＡとオブジェクトＢには並行性があり）、オブジェクトＡとオブジェクトＣとの間には実行逐次性があるとする。このとき、オブジェクトＡとオブジェクトＢを組み合わせて複合オブジェクトにすることはできない。一方、オブジェクトＡとオブジェクトＣを組み合わせて複合オブジェクトとすることは可能である。

【０１４９】

そこで、オブジェクトＡとオブジェクトＢについては、それらを組み合わせて用いるとしても、それらのオブジェクトＡ，Ｂをそのまま標準オブジェクトとして用いる。一方、オブジェクトＡとオブジェクトＣについて、それらを組み合わせて用いる場合は、それらのオブジェクトＡ，Ｃを複合オブジェクトを構成する

構成オブジェクトとすることができる。

【０１５０】

このように、実行逐次性を考慮して、標準オブジェクトと複合オブジェクトとを使い分けることで、システムのパフォーマンスを向上させることが可能となる。

【０１５１】

２－５－２　複合オブジェクトの動的リンク機構による実現

構成オブジェクトを標準オブジェクトと同様に、柔軟に追加や削除できるようにするために、構成オブジェクトの実現には、動的リンク機構が用いられる。

【０１５２】

図１０に、複合オブジェクトの構成を示したＯＭＴダイアグラムを示す。図１０に示すように、複合オブジェクト(Composite Object)は、任意の数の構成オブジェクト(Component Object)から成る。各構成オブジェクトはそれぞれのＯＩＤを持つ。複合オブジェクトもＯＩＤを一つ持つ。複合オブジェクトは一つの実行スレッド(Thread)を持つ。構成オブジェクトは、オブレット(Oblet)とエントリテーブル(EntryTable)から成る。オブレットの構造は図７及び図８を用いて説明した通りである。エントリテーブルからリンクテーブルに複数の参照がある。この構成は、2－4章で説明した動的共有ライブラリの構成の特殊な場合、すなわちオブレットがライブラリとして共有されていない場合の構成になっている。したがって、このオブレットも、動的共有ライブラリと同じ方式で初期化が行われる。

【０１５３】

また、標準オブジェクトも動的リンク機構を用いた構成になっている。これにより、標準オブジェクトの構成と複合オブジェクトの構成とを共通化でき、システムの処理を簡便化することができる。

【０１５４】

図１１に、標準オブジェクトの構成を示したＯＭＴダイアグラムを示す。図１１に示すように、標準オブジェクト(StandardObject)は、一つのＯＩＤと、一つの実行スレッド(Thread)とを持つ。また、標準オブジェクトは、オブレット(Obl

et)とエントリテーブル(EntryTable)を持つ。すなわち、標準オブジェクトの構成は、複合オブジェクトの特殊な場合、すなわち、複合オブジェクトが一つの構成オブジェクトのみを持っている場合と、同様な構造になっている。ただし、標準オブジェクトに対してＯＩＤは一つだけ与えられる。

【０１５５】

２－５－３　複合オブジェクトの初期化アルゴリズム

複合オブジェクトの初期化後の動作は標準オブジェクトと同様だが、初期化手続きについては、標準オブジェクトの場合とは異なる。図１２に複合オブジェクトの初期化手続きを示すとともに、図１３に複合オブジェクトに含まれる構成オブジェクトの初期化手続きを示す。

【０１５６】

標準オブジェクトは、プロローグメソッドが呼び出されて当該プロローグメソッドが実行されることで初期化手続きが行われる。複合オブジェクトもプロローグメソッドを持ち、当該プロローグメソッドが呼び出されて初期化手続きが行われるが、その初期化手続きは標準オブジェクトの場合とは異なり、図１２及び図１３に示すように行われる。

【０１５７】

複合オブジェクトの初期化手続きでは、図１２に示すように、先ず、ステップＳ１において、オブジェクト参照用テーブル（オブジェクト名からＯＩＤを参照するためのテーブル）を作成し、当該オブジェクト参照用テーブルを初期化する。このオブジェクト参照テーブルは、複合オブジェクトの内部のみで用いられる。しかし、その内容は、後述するように初期化手続きの最後でレジストリに登録するので、最終的には複合オブジェクトの外部からも参照可能となる。

【０１５８】

次に、ステップＳ２において、複合オブジェクトを構成する全ての構成オブジェクトについて、初期化手続きが完了しているか否かを判別する。初期化手続きが完了していない構成オブジェクトがある場合には、ステップＳ３へ進む。また、全ての構成オブジェクトの初期化手続きが完了しているならば、ステップＳ７へ進む。

【０１５９】

なお、複合オブジェクトの初期化手続きを実行する時点で、当該複合オブジェクトがどの構成オブジェクトを含むかは、「構成オブジェクトコンフィギュレーションファイル」に予め記述しておく。構成オブジェクトコンフィギュレーションファイルには、具体的には、構成オブジェクトの名前と、構成オブジェクトの持つエントリ数と、構成オブジェクトのプロローグメソッドのエントリポイントとを記述しておく。

【０１６０】

下記にコード３として、構成オブジェクトコンフィギュレーションファイルの一例を示す。なお、下記の例では、プログラミング言語Ｃ＋＋の構造体として構成オブジェクトコンフィギュレーションファイルを記述している。

【０１６１】

コード３：構成オブジェクトコンフィギュレーションファイル

```
1: componentObjectConfiguration [] = {
2:     /*      name,       entry_num,  prologue_method       */
3:    {"SchedulerPolicy",     8,      _SchedPolicyPrologue},
4:    {"SchedulerMechanism", 4,       _SchedMechanismPrologue},
5:    {"Timer",               3,      _TimerPrologue}
6: }
```

上記コード３は、「"SchedulerPolicy"」，「"SchedulerMechanism"」，「"Timer"」という名前を持つ３つの構成オブジェクトが、それぞれ8，4，3のエントリ数を持ち、それぞれ「_SchedPolicyPrologue」，「_SchedMechanismPrologue」，「_TimerPrologue」というプロローグメソッドを持っていることを示している。

【０１６２】

そして、ステップＳ３では、上述のような構成オブジェクトコンフィギュレーションファイルから、次に初期化手続きを行う構成オブジェクトの情報を獲得する。

【０１６３】

次に、ステップＳ４において、ステップＳ３で獲得した情報に基づいて、図１

３に示すように、構成オブジェクトの初期化手続きを行う。なお、構成オブジェクトの初期化手続きについては、後で図１３を参照して詳細に説明する。

【０１６４】

次に、ステップＳ５において、ステップＳ４での構成オブジェクトの初期化手続きが成功したか否かを判別する。構成オブジェクトの初期化手続きが成功しているならば、ステップＳ２へ戻って処理を繰り返す。一方、構成オブジェクトの初期化手続きに失敗しているならば、ステップＳ６へ進み、当該ステップＳ６において中間生成物の削除などのエラー処理を行った上で、ステップＳ２へ戻る。

【０１６５】

以上の処理を繰り返し、全ての構成オブジェクトの初期化手続きが完了したならば、上述したように、ステップＳ２からステップＳ７へと進む。そして、ステップＳ７において、オブジェクト参照用テーブルの内容をレジストリに登録し、複合オブジェクトの初期化手続きを終了する。

【０１６６】

なお、個々の複合オブジェクトに対して図１２に示すような初期化手続きを記述してもよいが、この初期化手続きはライブラリとして提供することが好ましい。殆どの複合オブジェクトは同様な初期手続きを実行するため、図１２に示すような初期化手続きをライブラリとして提供することにより、プログラミングを軽減することができる。

【０１６７】

つぎに、図１３を参照して、個々の構成オブジェクトの初期化手続きについて説明する。なお、以下に説明する初期化手続きは、上記ステップＳ４で実行されるものである。

【０１６８】

図１３に示すように、構成オブジェクトの初期化手続きでは、先ず、ステップＳ１１において、構成オブジェクトのプロローグメソッドを呼び出す。構成オブジェクトのプロローグメソッドは、２－３－２章のコード１で示した標準オブジェクトのものと同一のプログラミングスタイルで記述され、このプロローグメソッドにより外部インタフェースの登録が行われる。

【０１６９】

なお、構成オブジェクトのプロローグメソッドのプログラミングスタイルは標準オブジェクトの場合と同一だが、「InitEntryTable()」での処理が、対象が標準オブジェクトの場合と、対象が構成オブジェクトの場合とで若干異なる。すなわち、対象が標準オブジェクトの場合、「InitEntryTable()」はエントリテーブルの内容をオブジェクトに直接登録するが、対象が構成オブジェクトの場合、「InitEntryTable()」は、エントリテーブルの内容を一時的な記憶領域に格納し、その内容を後から構成オブジェクトに登録する。

【０１７０】

このように、「InitEntryTable()」での処理を、対象が標準オブジェクトの場合と、対象が構成オブジェクトの場合とで変えることで、プロローグメソッドのプログラミングスタイルを、標準オブジェクトのものと構成オブジェクトのものとで同一にすることができる。換言すれば、対象が標準オブジェクトの場合と、対象が構成オブジェクトの場合とで、「InitEntryTable()」での処理を若干異なるものとしているのは、プロローグメソッドのプログラミングスタイルを標準オブジェクトのものと構成オブジェクトのものとで同一にするための措置である。

【０１７１】

そして、ステップＳ１１で呼び出されたプロローグメソッドから制御が戻ったら、次に、ステップＳ１２において、プロローグメソッドの中で「InitEntryTable()」が正しく呼び出されたかをチェックをする。これは、エントリテーブルの内容が格納されている一時的な記憶領域を調べることで分かる。そして、「InitEntryTable()」が一度も呼び出されていなかったり、複数回呼び出されているような場合には、エラーを返す。

【０１７２】

一方、プロローグメソッドの中で「InitEntryTable()」が正しく呼び出されている場合には、ステップＳ１３に進み、一時的な記憶領域に格納されているエントリテーブルの内容を獲得する。

【０１７３】

次に、ステップＳ１４において、一時的な記憶領域に格納されているエントリ

テーブルの内容を獲得できたかをチェックする。そして、初期化中の構成オブジェクトに対応したエントリーテーブルの内容が一時的な記憶領域に格納されておらず、エントリーテーブルの内容を獲得できなかった場合には、エラーを返す。

【０１７４】

一方、エントリーテーブルの内容を獲得できた場合には、ステップＳ１５に進み、「構成オブジェクト記述子」を作成する。構成オブジェクト記述子は、構成オブジェクトのエントリテーブル等が格納される構造体である。なお、構成オブジェクト記述子については、２－５－４章で詳述する。

【０１７５】

次に、ステップＳ１６において、構成オブジェクト記述子にエントリテーブルを登録する。具体的には、先ず、エントリテーブルを作成し、当該エントリーテーブルに、一時的な記憶領域に格納されているエントリーテーブルの内容をコピーする。その後、このエントリテーブルを構成オブジェクト記述子に登録する。

【０１７６】

次に、ステップＳ１７において、構成オブジェクトのＯＩＤを生成する。

【０１７７】

次に、ステップＳ１８において、オブジェクト参照用リストにオブジェクト名とＯＩＤを登録する。

【０１７８】

以上の処理で構成オブジェクトの初期化手続きが完了し、図１２に示した複合オブジェクトの初期化手続きにおけるステップＳ４の処理が完了したこととなる。

【０１７９】

なお、以上のような構成オブジェクトの初期化手続きの最中にエラーが発生した場合は、図１２に示した複合オブジェクトの初期化手続きにおいて、ステップＳ６へ進むこととなる。

【０１８０】

２－５－４　オブジェクト記述子

オブジェクトの情報を格納する記述子について、図１４のＯＭＴダイアグラム

を参照して説明する。

【０１８１】

図１４に示すように、「オブジェクト記述子」「標準オブジェクト記述子」「構成オブジェクト記述子」の３種類のクラスが定義されている。クラス「オブジェクト記述子」は抽象クラスであり、クラス「標準オブジェクト記述子」及びクラス「構成オブジェクト記述子」は、クラス「オブジェクト記述子」のサブクラスになっている。

【０１８２】

そして、オブジェクト記述子は、「オブジェクト型」「OID」を持っている。オブジェクト記述子の「オブジェクト型」には、対応するオブジェクトが標準オブジェクトと構成オブジェクトのどちらであるかを示すフラグが格納され、オブジェクト記述子の「OID」には、対応するオブジェクトのＯＩＤが入る。

【０１８３】

標準オブジェクト記述子は、標準オブジェクトに関する情報を格納するための構造体であり、一つの標準オブジェクトは、標準オブジェクト記述子を一つ保持する。また、複合オブジェクトも、標準オブジェクト記述子を一つ保持する。

【０１８４】

標準オブジェクト記述子は、「実行スレッド」「メモリスペースID」「ヒープリスト」を持っている。標準オブジェクト記述子の「メモリスペースID」には、対応する標準オブジェクトが使用するメモリスペースの識別子が格納され、「ヒープリスト」には、対応する標準オブジェクトが使用するヒープメモリ領域のリストが格納される。

【０１８５】

また、標準オブジェクト記述子の「実行スレッド」には、対応する標準オブジェクトの実行スレッドに関する情報として、「エントリテーブル」「実行スタック」「実行モード」「割込みレベル」が格納される。ここで、「実行モード」は、対応するオブジェクトが特権命令モードであるか否かを示し、「割込みレベル」は、割込みの優先順位を示す。

【０１８６】

構成オブジェクト記述子は、構成オブジェクトに関する情報を格納するための構造体であり、一つの構成オブジェクトは、構成オブジェクト記述子を一つ保持する。

【０１８７】

構成オブジェクト記述子は、対応する構成オブジェクトが属している複合オブジェクトに対応した「標準オブジェクト記述子」と、対応する構成オブジェクトの「エントリテーブル」と、リンクテーブルなどを含む「オブレット」とを持っている。なお、オブレットの構成は図７及び図８を用いて説明した通りである。そして、この構成オブジェクト記述子には、２－５－３章で図１３を参照して説明したように、構成オブジェクトの初期化手続き時に値が書き込まれる。

【０１８８】

２－５－５　複合オブジェクトを考慮したメッセージ通信機構

構成オブジェクト間でメッセージ通信を行うときのプログラミングスタイルや、構成オブジェクトと標準オブジェクトとの間でメッセージ通信を行うときのプログラミングスタイルは、いずれも標準オブジェクト間でメッセージ通信を行うときのプログラミングスタイルと同一である。すなわち、構成オブジェクト間でメッセージ通信を行うときや、構成オブジェクトと標準オブジェクトとの間でメッセージ通信を行うときも、２－３－１章で示したＡＰＩを用いてプログラミングが行われる。ただし、構成オブジェクト間でのメッセージ通信は、２－５－１章で示したように、二つのオブジェクト間で実行逐次性があることを考慮する必要がある。

【０１８９】

以上のように、プログラミングスタイルの互換性を保つには、複数の構成オブジェクトから成る複合オブジェクトを考慮する必要がある。

【０１９０】

以下、複合オブジェクトを考慮して、プログラミングスタイルの互換性を保ったままメッセージ通信を行えるようにしたメッセージ通信機構について、詳細に説明する。なお、以下の説明では、メッセージ送信側のオブジェクトが構成オブ

ジェクトの場合と、メッセージ送信側のオブジェクトが標準オブジェクトの場合とに分けて説明する。

【０１９１】

（１）メッセージ送信側のオブジェクトが構成オブジェクトの場合

メッセージ送信側のオブジェクトが構成オブジェクトの場合、メッセージの処理は複合オブジェクトにリンクされているライブラリの中で開始される。メッセージ送信側のオブジェクトが構成オブジェクトの場合に、メッセージ送信用のＡＰＩ「Send()」が発行されたときの手続きを図１５を参照して説明する。

【０１９２】

先ず、ステップＳ２１において、「Send()」の引数として与えられる受信オブジェクトのＯＩＤからオブジェクト記述子を獲得する。なお、オブジェクト記述子とＯＩＤとの対応関係を記述した「オブジェクトテーブル」を予め作成しておき、ＯＩＤからオブジェクト記述子を獲得する際は、このオブジェクトテーブルを参照する。

【０１９３】

次に、ステップＳ２２において、オブジェクト記述子のオブジェクト型を調べる。そして、オブジェクト型が構成オブジェクトを示している場合（すなわち、受信オブジェクトが構成オブジェクトの場合）は、ステップＳ２３に進む。

【０１９４】

ステップＳ２３では、構成オブジェクト記述子の「実行スレッド」から、対応する構成オブジェクトの「エントリテーブル」を獲得する。

【０１９５】

次に、ステップＳ２４において、「Send()」の引数として与えられるメッセージセレクタに対応したエントリを、ステップＳ２３で獲得した「エントリテーブル」から獲得する。

【０１９６】

次に、ステップＳ２５において、ステップＳ２４で正しくエントリが獲得できたかをチェックし、正しくエントリが獲得できなかった場合にはエラーを返す。正しくエントリが獲得できた場合には、ステップＳ２６へ進み、ステップＳ２４

で獲得したエントリに直接ジャンプして処理を行う。これにより、送信オブジェクトから受信オブジェクトへ制御が移り、メッセージ通信が行われたこととなる。すなわち、送信オブジェクト及び受信オブジェクトが共に構成オブジェクトの場合には、実行スレッドの切り替えは起こらずに、関数呼び出しと同様な制御の流れで、オブジェクト間通信が行われる。

【０１９７】

また、ステップＳ２２において、オブジェクト記述子のオブジェクト型が標準オブジェクトを示している場合（すなわち、受信オブジェクトが標準オブジェクトの場合）には、ステップＳ２７に進む。

【０１９８】

ステップＳ２７では、標準オブジェクト記述子の「実行スレッド」を獲得する。換言すれば、このステップＳ２７では、受信オブジェクトに対応した「実行スレッド」を獲得する。

【０１９９】

次に、ステップＳ２８において、ステップＳ２７で獲得した「実行スレッド」から、対応する標準オブジェクトの「エントリテーブル」を獲得する。

【０２００】

次に、ステップＳ２９において、「Send()」の引数として与えられるメッセージセレクタに対応したエントリを、ステップＳ２８で獲得した「エントリテーブル」から獲得する。

【０２０１】

次に、ステップＳ３０において、ステップＳ２９で正しくエントリが獲得できたかをチェックし、正しくエントリが獲得できなかった場合にはエラーを返す。正しくエントリが獲得できた場合には、ステップＳ３１へ進む。

【０２０２】

ステップＳ３１では、図３や図４で示したようなメッセージ通信の処理を行う。このとき、図３や図４を参照して説明したように、メッセージキュー操作やオブジェクト「スケジューリングポリシ」の呼び出し等の処理を行い、最終的に受信オブジェクトへの実行スレッドの切り替えが起こる。なお、受信オブジェクト

がオブジェクト「スケジューリングポリシ」の場合にも、基本的にはこの手続きが用いられるが、その場合にはスケジューリングポリシの内部で例外的な処理が行われる。

【０２０３】

以上が、構成オブジェクトがメッセージ送信用のＡＰＩ「Send()」を発行したときに行われる手続きである。

【０２０４】

以上の説明から分かるように、メッセージ通信が構成オブジェクト間で行われる場合は、メッセージキュー操作、オブジェクト「スケジューリングポリシ」の呼び出し、実行スレッドの切り替え等の処理（すなわちステップＳ３１での処理）が省略されるので、通信コストが大幅に低減される。

【０２０５】

（２）メッセージ送信側のオブジェクトが標準オブジェクトの場合

メッセージ送信側のオブジェクトが標準オブジェクトの場合、メッセージの処理はオブジェクト「メッセージハンドラ」で開始される。メッセージ送信側のオブジェクトが標準オブジェクトの場合に、メッセージ送信用のＡＰＩ「Send()」が発行されたときの手続きを図１６を参照して説明する。

【０２０６】

先ず、ステップＳ４１において、「Send()」の引数として与えられる受信オブジェクトのＯＩＤからオブジェクト記述子を獲得する。なお、オブジェクト記述子とＯＩＤとの対応関係を記述した「オブジェクトテーブル」を予め作成しておき、ＯＩＤからオブジェクト記述子を獲得する際は、このオブジェクトテーブルを参照する。

【０２０７】

次に、ステップＳ４２において、オブジェクト記述子のオブジェクト型を調べる。そして、そのオブジェクト型が構成オブジェクトを示している場合（すなわち、受信オブジェクトが構成オブジェクトの場合）は、ステップＳ４３に進む。また、そのオブジェクト型が標準オブジェクトを示している場合（すなわち、受信オブジェクトが標準オブジェクトの場合）は、ステップＳ４４に進む。

【０２０８】

ステップＳ４３では、構成オブジェクト記述子経由で、受信オブジェクトが属する複合オブジェクトの標準オブジェクト記述子を得て、その標準オブジェクト記述子から、複合オブジェクトの「実行スレッド」を獲得する。その後、ステップＳ４５へ進む。

【０２０９】

一方、ステップＳ４４では、受信オブジェクトの標準オブジェクト記述子を得て、その標準オブジェクト記述子から、受信オブジェクトの「実行スレッド」を獲得する。その後、ステップＳ４５へ進む。

【０２１０】

ステップＳ４５では、ステップＳ４３又はステップＳ４４で獲得した「実行スレッド」から、対応する構成オブジェクトの「エントリテーブル」を獲得する。

【０２１１】

次に、ステップＳ４６において、「Send()」の引数として与えられるメッセージセレクタに対応したエントリを、ステップＳ４５で獲得した「エントリテーブル」から獲得する。

【０２１２】

次に、ステップＳ４７において、ステップＳ４６で正しくエントリが獲得できたかをチェックし、正しくエントリが獲得できなかった場合にはエラーを返す。正しくエントリが獲得できた場合には、ステップＳ４８へ進む。

【０２１３】

ステップＳ４８では、図３や図４で示したようなメッセージ通信の処理を行う。このとき、図３や図４を参照して説明したように、メッセージキュー操作やオブジェクト「スケジューリングポリシ」の呼び出し等の処理を行い、最終的に受信オブジェクトへの実行スレッドの切り替えが起こる。なお、受信オブジェクトがオブジェクト「スケジューリングポリシ」の場合にも、基本的にはこの手続きが用いられるが、その場合にはスケジューリングポリシの内部で例外的な処理が行われる。

【０２１４】

以上が、標準オブジェクトがメッセージ送信用のＡＰＩ「Send()」を発行したときに行われる手続きである。なお、受信オブジェクトと送信オブジェクトの両者が標準オブジェクトの場合の手続きは、ステップＳ４２においてオブジェクト記述子のチェックを行うこと以外、複合オブジェクトを導入する以前の手続きと同様なものとなっている。

【０２１５】

２－５－６　複合オブジェクトの例

本章では、図２で示したオブジェクト指向オペレーティングシステムにおいて、オペレーティングシステムサービスを提供するシステムオブジェクトの一部を複合オブジェクトとして実現し、システムオブジェクト間の通信コストを低減して、オペレーティングシステム全体の実行性能を向上させる例を示す。

【０２１６】

ここでは、図１７に示すように、複合オブジェクト「システムコア」を構成する。複合オブジェクト「システムコア」は、「タイマ」「メモリスイッチャ」「スケジューリングポリシ」「スケジューリング機構」「割込みベクタ」を構成オブジェクトとして持つ。

【０２１７】

図３や図４で示したように、オブジェクト間のメッセージ通信の手続きにおいては、オブジェクト「メモリスイッチャ」、オブジェクト「スケジュリングポリシ」、オブジェクト「スケジューリング機構」、オブジェクト「タイマ」の間で頻繁なメッセージ交換が行われるが、これらのオブジェクト間には実行逐次性がある。したがって、これらのオブジェクトは、複合オブジェクト「システムコア」を構成する構成オブジェクトして定義した方が、メッセージ通信機構の性能が向上し、結果的にシステム全体の実行性能が向上する。

【０２１８】

また、オブジェクト「割込みベクタ」は、割込み処理の時にオブジェクト「スケジューリングポリシ」やオブジェクト「スケジューリング機構」とメッセージ交換を行うが、これら三者の間には実行逐次性がある。したがって、これらのオ

ブジェクトは、複合オブジェクト「システムコア」を構成する構成オブジェクトして定義した方が、割り込み処理の性能が向上し、結果的にシステム全体の実行性能が向上する。

【０２１９】

なお、複合オブジェクトに組み込む構成オブジェクトの種類は、「構成オブジェクトコンフィギュレーションファイル」に記述しておくことで予め定義することが可能である。すなわち、オペレーティングシステムのサービス提供部分のうち、複合オブジェクトとして定義されたサービス提供部分に組み込む構成オブジェクトの種類は、２－５－３章のコード３で示したように、「構成オブジェクトコンフィギュレーションファイル」に記述しておけばよい。また、複合オブジェクトに組み込む構成オブジェクトは、動的に変更したり交換したりすることも可能である。

【０２２０】

したがって、オペレーティングシステムのサービス提供部分のうち、複合オブジェクトとして定義されたサービス提供部分は、所望の動作を行うように、当該複合オブジェクトを構成する構成オブジェクトを定義したり、或いは、必要に応じて後から動的に、当該複合オブジェクトを構成する構成オブジェクトを変更したり交換したりすることが可能である。なお、構成オブジェクトを動的に変更や交換する方法については、２－６章で説明する。

【０２２１】

２－６　構成オブジェクトの動的な複合化と分離

標準オブジェクトを複合オブジェクト中の構成オブジェクトにすることを、オブジェクトの「複合化」と呼ぶ。逆に、複合オブジェクト中の構成オブジェクトを複合オブジェクトから抜き出して標準オブジェクトにすることを、オブジェクトの「分離」と呼ぶ。本章では、複合化やと分離を動的に(すなわち、オペレーティングシステムの実行中に)行う方法について述べる。動的な複合化や分離を行い、構成オブジェクトの種類の変更や交換を行うことにより、オペレーティングシステムの動作は柔軟に変更可能である。

【０２２２】

２－６－１　複合化及び分離に用いられるＡＰＩ

オブジェクトの複合化や分離に用いられるＡＰＩについて、具体的な例を挙げて説明する。

【０２２３】

オブジェクトの複合化には、ＡＰＩ「AddComponent()」が用いられる。また、オブジェクトの分離には、ＡＰＩ「RemoveComponent()」が用いられる。これらのＡＰＩは、複合オブジェクトにリンクされるライブラリ中に定義されており、複合オブジェクトで用いることが可能である。以下、これらのＡＰＩについて詳細に説明する。

【０２２４】

```
sError AddComponent(in char* object_name, in longword entry_num,
                    in Entry prologue_entry, out OID object_oid);
```

このＡＰＩは、標準オブジェクトとして定義されたオブジェクトを、このＡＰＩを発行した複合オブジェクトの構成オブジェクトにする。このＡＰＩにおいて、入力引数「object_name」には、構成オブジェクトとするオブジェクトの名称を設定する。複合化されたオブジェクトの初期化後に、この入力引数「object_name」に設定した名称が、レジストリに登録される。この名称は、このＡＰＩにより複合化されたオブジェクトを他のオブジェクトが参照するときに用いられる。また、入力引数「entry_num」には、複合化されるオブジェクトのエントリ数を設定する。また、入力引数「prologue_entry」には、複合化されるオブジェクトが初期化されるときに呼ばれるプロローグメソッドのエントリポイントを設定する。また、このＡＰＩは戻り値として、複合化されて生成された構成オブジェクトのＯＩＤを、出力引数「object_oid」に格納する。

【０２２５】

```
sError RemoveComponent(in char* object_name, out OID object_oid);
```

このＡＰＩは、このＡＰＩを発行した複合オブジェクト内の構成オブジェクトを、標準オブジェクトとして分離する。ここで、入力引数「object_name」には、複合オブジェクトから分離するオブジェクトの名称を設定する。すなわち、こ

のＡＰＩは、入力引数「object_name」で参照されるオブジェクトを、このＡＰＩを発行した複合オブジェクトから分離し、標準オブジェクトとして独立させる。また、このＡＰＩは戻り値として、複合オブジェクトから分離されて生成された標準オブジェクトのＯＩＤを、出力引数「object_oid」に格納する。

【０２２６】

なお、「AddComponent()」や「RemoveComponent()」が発行された場合、オブジェクトの複合化や分離の手続きが完了した時点でオブジェクトの状態はリセットされ、再び初期化手続きが行われる。

【０２２７】

以上のようなＡＰＩを用いて、既存の２つの標準オブジェクトを複合化し、１つの複合オブジェクトの構成オブジェクトにするプログラムの例を、下記にコード４として示す。コード４のプログラムでは、オブジェクト「SystemCore」にオブジェクト「ObjectA」及びオブジェクト「ObjectB」を組み込む。このとき、オブジェクト「ObjectC」のメソッド「AddAandB()」が、オブジェクト「SystemCore」を呼び出して、これらのオブジェクトを登録させる。

【０２２８】

コード４：複合化のプログラミング例

```
 1: void SystemCore::Add (char* name, number, prologue)
 2: {
 3:       AddComponent (name, number, prologue)
 4: }
 5:
 6: void ObjectC::AddAandB ()
 7: {
 8:       #define NAME_OBJECT_A "ObjectA"
 9:       #define NAME_OBJECT_B "ObjectB"
10:
11:     OID       objectA, objectB, systemCore;
12:
```

```
13:     struct SystemCoreAddMsg {
14:         char     name [16];
15:         longword number;
16:         Entry    entry;
17:     } msgA, msgB;
18:
19:     FindOID ("SystemCore", &systemCore);
20:     FindOID (NAME_OBJECT_A, &objectA);
21:     FindOID (NAME_OBJECT_B, &objectB);
22:     strcpy (msgA.name, NAME_OBJECT_A);
23:     strcpy (msgB.name, NAME_OBJECT_B);
24:     msgA.number = FindEntryNumber (objectA);
25:     msgB.number = FindEntryNumber (objectB);
26:     msgA.entry  = FindPrologue (objectA);
27:     msgB.entry  = FindPrologue (objectB);
28:
29:     Send (systemCore, INDEX_ADD, &msgA);
30:     Send (systemCore, INDEX_ADD, &msgB);
31: }
```

上記コード４のプログラム例において、１～３行目はオブジェクト「SystemCore」のメソッド「Add()」であり、最終的に構成オブジェクトを追加するために、ＡＰＩ「AddComponent() 」を呼び出している。

【０２２９】

このメソッド「Add()」を呼び出すのが、６行目以下に記載されたメソッド「ObjectC::AddAandB()」である。

【０２３０】

メソッド「ObjectC::AddAandB()」では、１９行目において、後で行われるメッセージ送信に備えて、オブジェクト「SystemCore」を検索している。また、２０行目でオブジェクト「ObjectA」を検索し、２１行目でオブジェクト「ObjectB

」を検索している。

【０２３１】

２２～２７行目では、１３～１７行目で定義されている構造体「SystemCoreAddMsg」に値をセットしている。この構造体「SystemCoreAddMsg」の値は、メッセージ送信の過程で、オブジェクト「SystemCore」のメソッド「Add()」の３つの引数にそれぞれ渡される。

【０２３２】

２４，２５行目で用いられているメソッド「FindEntryNumber()」は、引数で渡されるＯＩＤを持つオブジェクトのエントリ数を獲得する。すなわち、２４行目では、メソッド「FindEntryNumber()」により、オブジェクト「ObjectA」のエントリ数が獲得され、２５行目では、メソッド「FindEntryNumber()」により、オブジェクト「ObjectB」のエントリ数が獲得されている。

【０２３３】

２６，２７行目で用いられているメソッド「FindPrologue()」は、引数で渡されるＯＩＤを持つオブジェクトのプロローグメソッドのエントリポイントを獲得する。すなわち、２６行目では、メソッド「FindPrologue()」により、オブジェクト「ObjectA」プロローグメソッドのエントリポイントが獲得され、２７行目では、メソッド「FindPrologue()」により、オブジェクト「ObjectB」プロローグメソッドのエントリポイントが獲得されている。

【０２３４】

２９行目では、オブジェクト「SystemCore」のメソッド「Add()」に対するメッセージが、オブジェクト「ObjectA」に関するパラメータと共に送信されている。ここで、引数「INDEX_ADD」は、オブジェクト「SystemCore」のメソッド「Add()」に対応するメソッドセレクタである。

【０２３５】

３０行目では、オブジェクト「SystemCore」のメソッド「Add()」に対するメッセージが、オブジェクト「ObjectB」に関するパラメータと共に送信されている。ここで、引数「INDEX_ADD」は、オブジェクト「SystemCore」のメソッド「Add()」に対応するメソッドセレクタである。

【０２３６】

なお、上記のプログラム例では、２つのオブジェクト「ObjectA」「ObjectB」を複合化する部分だけを示したが、オブジェクト定義、複合化、分離、オブジェクト削除、オブジェクト再定義、再複合化というサイクルをくり返すと、オペレーティングシステムの動作を柔軟に変化させていくようなことも可能である。

【０２３７】

２－６－２　複合化アルゴリズム及び分離アルゴリズム

ＡＰＩ「AddComponent()」が発行されたときに実行される複合化のアルゴリズムと、ＡＰＩ「RemoveComponent()」が発行されたときに実行される分離のアルゴリズムとについて説明する。

【０２３８】

（１）「AddComponent()」

「AddComponent()」が発行されると、図１３に示した構成オブジェクトの初期化手続きが呼ばれる。この手続きについては、２－５－３章で説明した通りである。ただし、２－５－３章では、複合オブジェクトの初期化手続きを行う場合を説明しており、初期化対象の複合オブジェクト自身が、既存の「構成オブジェクトコンフィギュレーションファイル」を読み込んで、構成オブジェクトの情報を得ていた。これに対して、「AddComponent()」が発行された場合は、「AddComponent()」の引数として、構成オブジェクトの情報が与えられる。また、新しく生成された構成オブジェクトのＯＩＤは、「AddComponent()」の出力引数「object_oid」に格納される。

【０２３９】

（２）「RemoveComponent()」

「RemoveComponent()」が発行されたときに実行される分離のアルゴリズムについては、図１８を参照して説明する。

【０２４０】

「RemoveComponent()」が発行されると、先ず、ステップＳ５１において、「RemoveComponent()」の入力引数「object_name」に設定されたオブジェクト名から、分離対象の構成オブジェクトのＯＩＤを獲得する。

【０２４１】

次に、ステップＳ５２において、複合オブジェクトのオブジェクト参照用リストから、分離対象の構成オブジェクトのオブジェクト名とＯＩＤを削除する。

【０２４２】

次に、ステップＳ５３において、分離対象の構成オブジェクトのＯＩＤを削除する。

【０２４３】

次に、ステップＳ５４において、分離対象の構成オブジェクトのエピローグメソッドを呼び出す。エピローグメソッドは、オブジェクトの削除時に呼ばれるメソッドであり、例えば、不必要なデータ領域の解放を行う。

【０２４４】

次に、ステップＳ５５において、分離対象の構成オブジェクトのエントリテーブルを削除する。

【０２４５】

次に、ステップＳ５６において、分離対象の構成オブジェクトに対応した構成オブジェクト記述子を削除する。

【０２４６】

以上で、分離の動作を終了し、「RemoveComponent()」の入力引数「object_name」にオブジェクト名が設定されていた構成オブジェクトが、複合オブジェクトから削除される。

【０２４７】

２－７　複合オブジェクトの導入による効果

複合オブジェクトを導入することにより、オブジェクト間のメッセージ通信のコストを低減することができる。しかも、オペレーティングシステムのサービス提供部分を複合オブジェクトとすることにより、オペレーティングシステムのサービス提供部分を動的に追加したり削除したりすることも可能となる。

【０２４８】

すなわち、複合オブジェクトを導入することにより、オペレーティングシステムの柔軟性を保った上で、オブジェクト間通信コストを低減して、オペレーティ

ングシステム全体の実行性能を向上することができる。

【０２４９】

【発明の効果】

以上詳細に説明したように、本発明によれば、システムコンフィギュレーションの柔軟性等、オブジェクト指向オペレーティングシステムの優れた特徴を保ちつつ、システム全体の実行性能を向上することができる。

【図面の簡単な説明】

【図１】

本発明を適用したテレビジョン受信装置の構成例を示す図である。

【図２】

オペレーティングシステムを構成するオブジェクト群の一例を示す図である。

【図３】

オペレーティングシステム上で動作する２つのアプリケーションオブジェクトの間でメッセージ通信が行われるときの実行遷移の一例を示す図である。

【図４】

システムオブジェクト間でメッセージ通信を行うときの実行遷移の一例を示す図である。

【図５】

メッセージ通信用のＡＰＩを利用した場合の実行遷移の一例を示す図である。

【図６】

エントリテーブルの一例を示す図である。

【図７】

オブレットの構造をＯＭＴダイアグラムにより示した図である。

【図８】

オブジェクトＡとオブジェクトＢで、オブレットＣを使った動的共有ライブラリを使用した例を示す図である。

【図９】

２つの標準オブジェクト「オブジェクトＡ」「オブジェクトＢ」から、複合オブジェクト「オブジェクトＣ」を構成した例を示す図である。

【図１０】

複合オブジェクトの構造をＯＭＴダイアグラムにより示した図である。

【図１１】

標準オブジェクトの構造をＯＭＴダイアグラムにより示した図である。

【図１２】

複合オブジェクトの初期化手続きの処理手順を示す図である。

【図１３】

複合オブジェクトに含まれる構成オブジェクトの初期化手続きの処理手順を示す図である。

【図１４】

オブジェクトの情報を格納するオブジェクト記述子をＯＭＴダイアグラムにより示した図である。

【図１５】

メッセージ送信側のオブジェクトが構成オブジェクトの場合に、メッセージ送信用のＡＰＩ「Send()」が発行されたときに実行される手続きの処理手順を示す図である。

【図１６】

メッセージ送信側のオブジェクトが標準オブジェクトの場合に、メッセージ送信用のＡＰＩ「Send()」が発行されたときに実行される手続きの処理手順を示す図である。

【図１７】

オペレーティングシステムのサービスを提供するシステムオブジェクトの一部を複合オブジェクトとして実現した例を示す図である。

【図１８】

オブジェクトの分離用のＡＰＩ「RemoveComponent()」が発行されたときに実行される手続きの処理手順を示す図である。

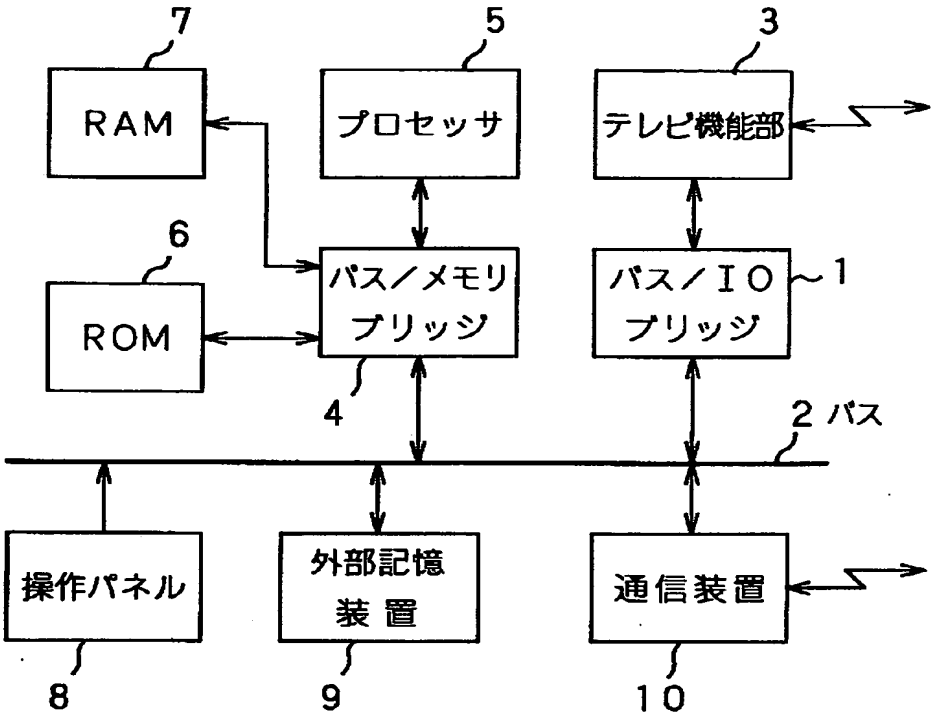
【符号の説明】

１　バス／ＩＯブリッジ、　２　バス、　３　テレビ機能部、　４　バス／メモリブリッジ、　５　プロセッサ、　６　ＲＯＭ（Read Only Memory）、　７

ＲＡＭ（Random Access Memory）、　８　操作パネル、　９　外部記憶装置、１０　通信装置

【書類名】　図面

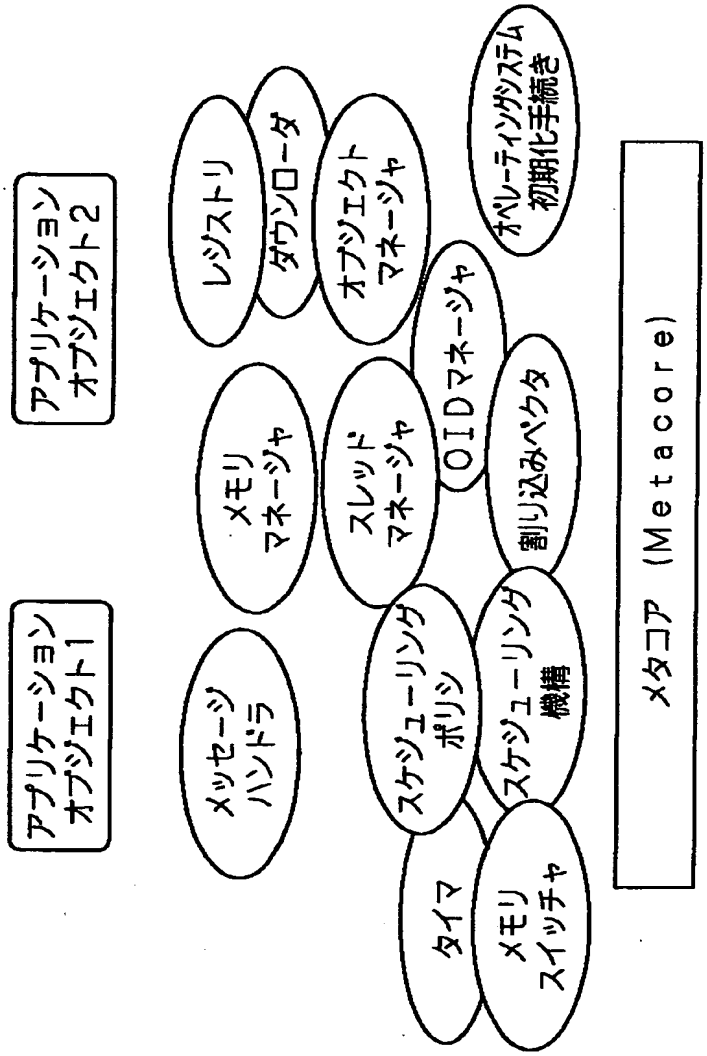
【図１】



テレビジョン受信装置の概略構成

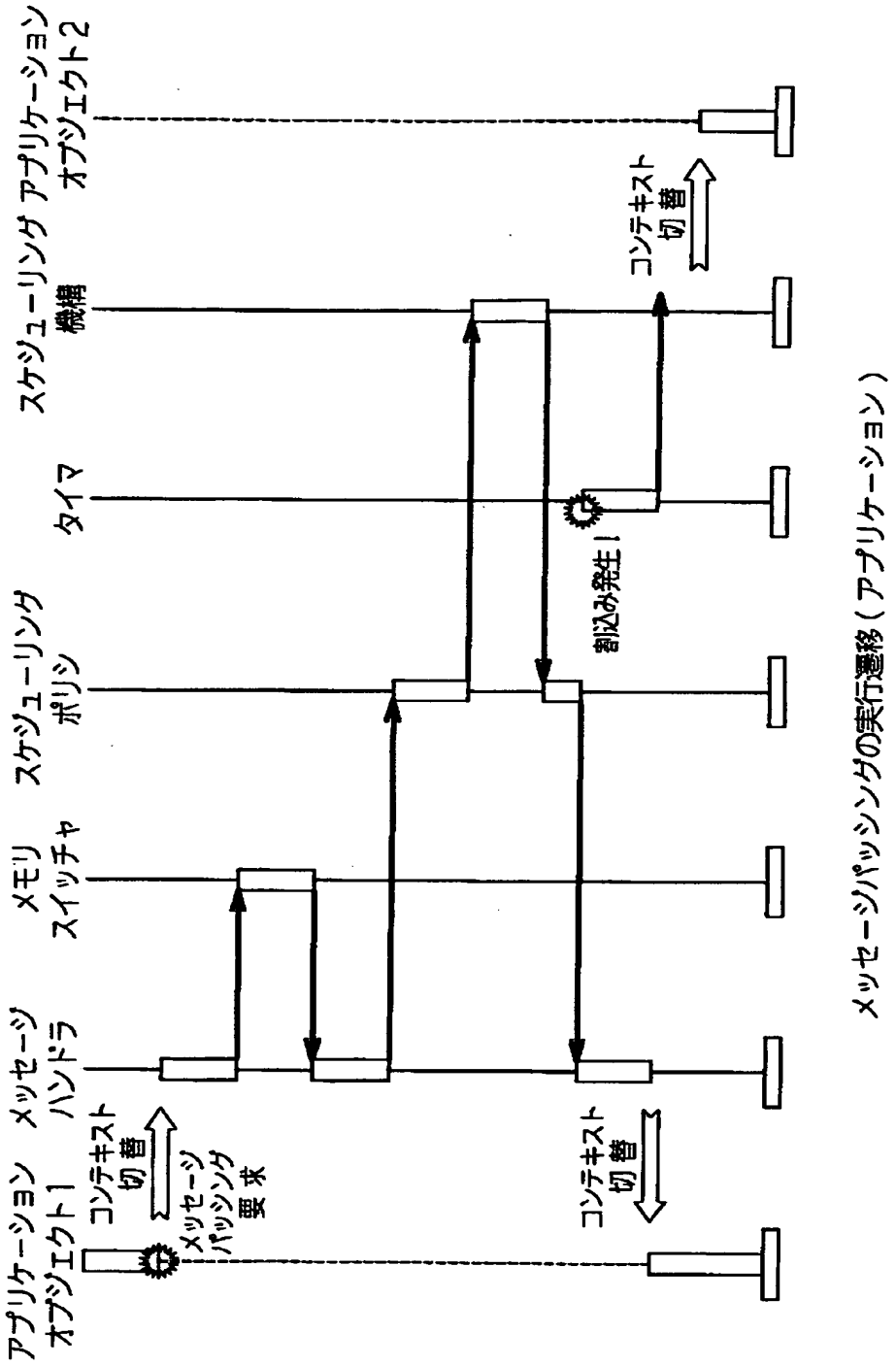
出証特２０００－３００３２７８

【図２】

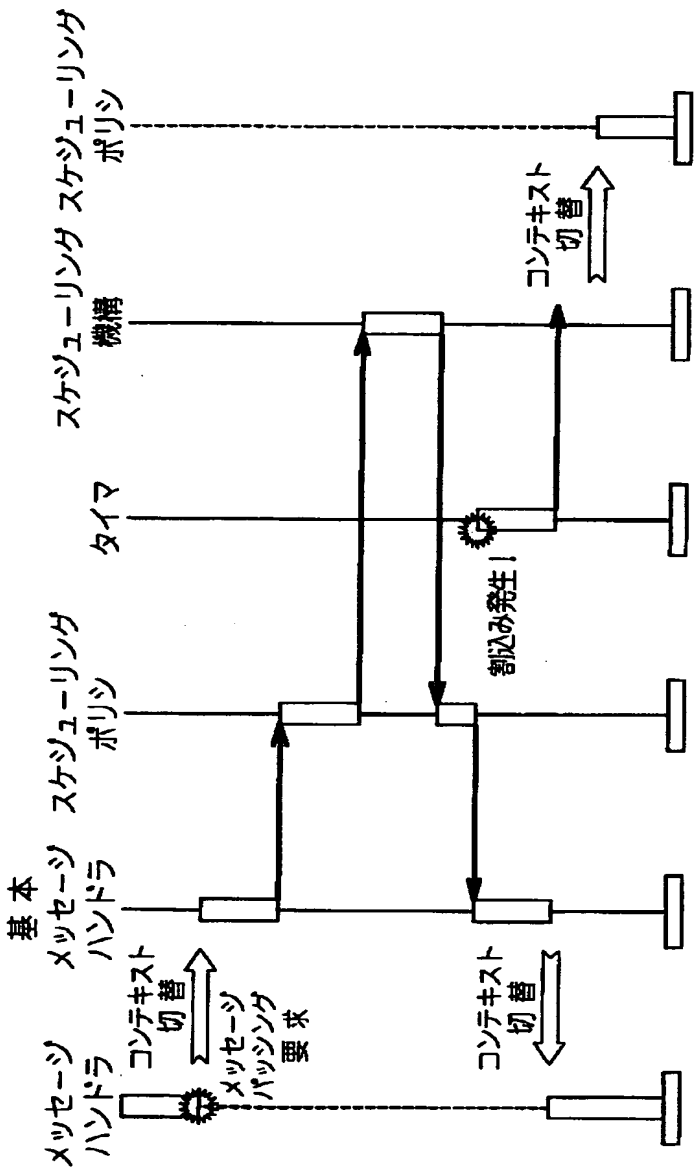


オペレーティングシステムの構成

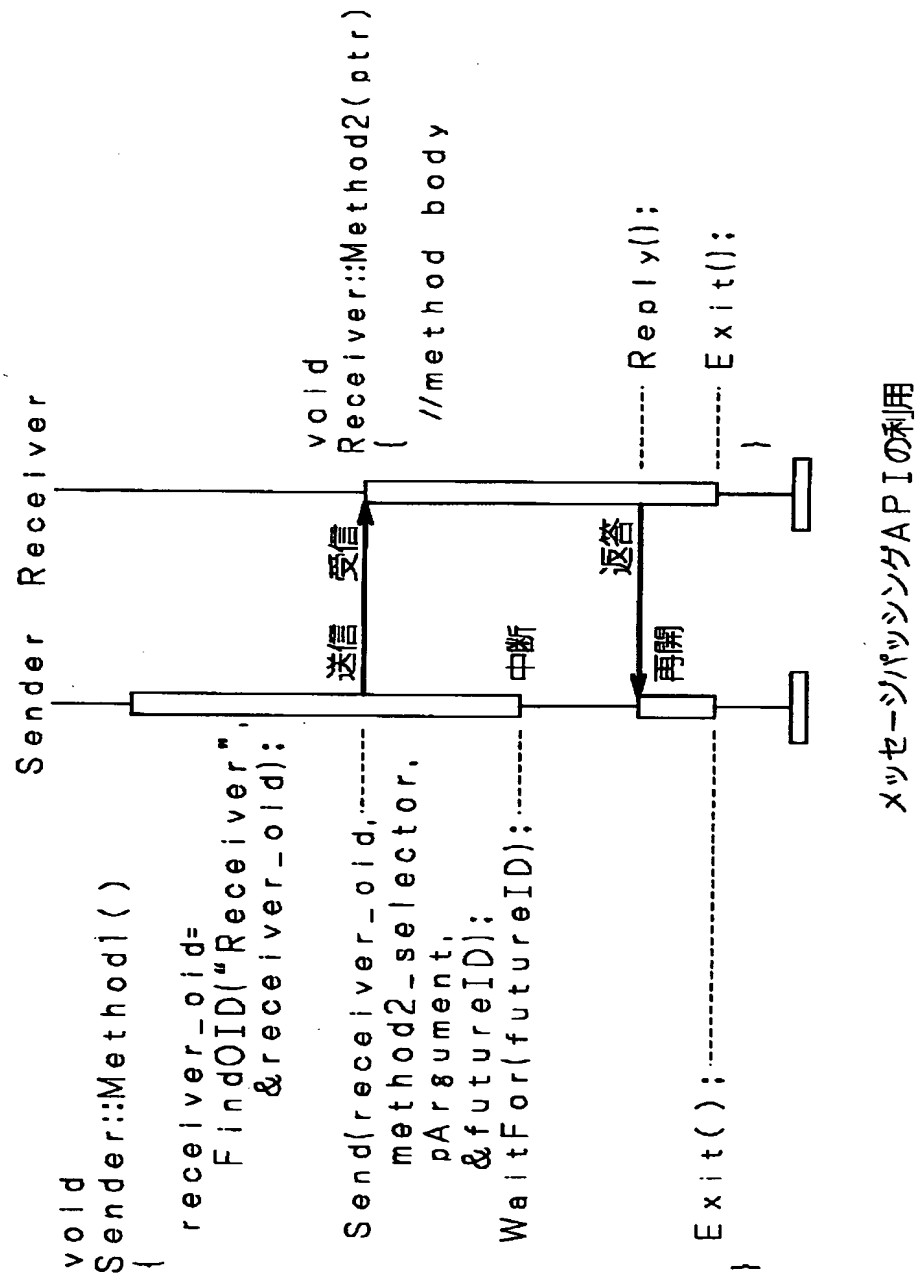
【図３】


アプリケーション オブジェクト1
メッセージ ハンドラ
メモリ スイッチャ
スケジューリング ポリシ
タイマ
スケジューリング 機構
アプリケーション オブジェクト2
コンテキスト 切替
メッセージ パッシング 要求
コンテキスト 切替
割込み発生！
コンテキスト 切替
メッセージパッシングの実行遷移（アプリケーション）

【図４】


メッセージパッシングの実行遷移（システム）
基本メッセージハンドラ
スケジューリング ポリシ
タイマ
スケジューリング 機構
スケジューリング ポリシ
メッセージハンドラ
コンテキスト切替
メッセージパッシング要求
コンテキスト切替
割込み発生！
コンテキスト切替

【図５】

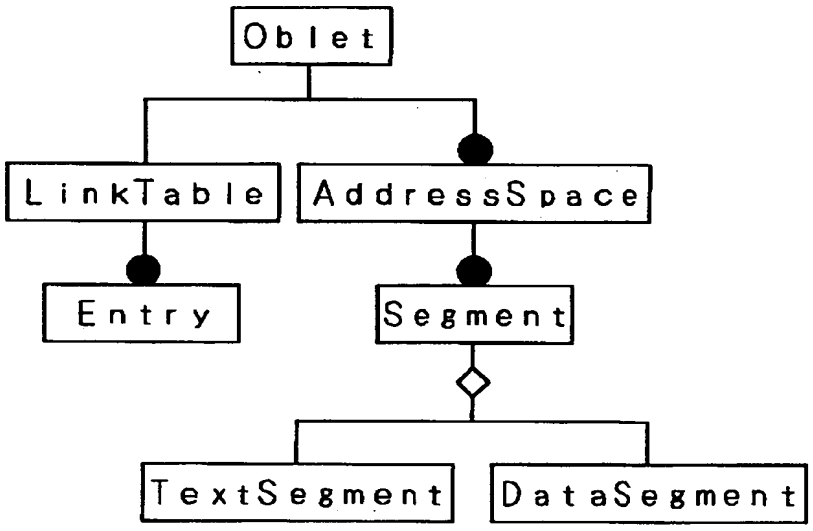

Sender Receiver
void
Sender::Method1()
{
receiver_oid=
FindOID("Receiver",
&receiver_oid);
Send(receiver_oid,
method2_selector,
pArgument,
&futureID);
WaitFor(futureID);
Exit();
}
void
Receiver::Method2(ptr)
{
//method body
Reply();
Exit();
}
送信
受信
中断
返答
再開
メッセージパッシングAPIの利用

【図６】

メッセージ セレクタ	エントリ （メソッドへのポインタ）
selector_1	entry_1
selector_2	entry_2
・・・・・・・・・	・・・・・・・
selector_n	entry_n

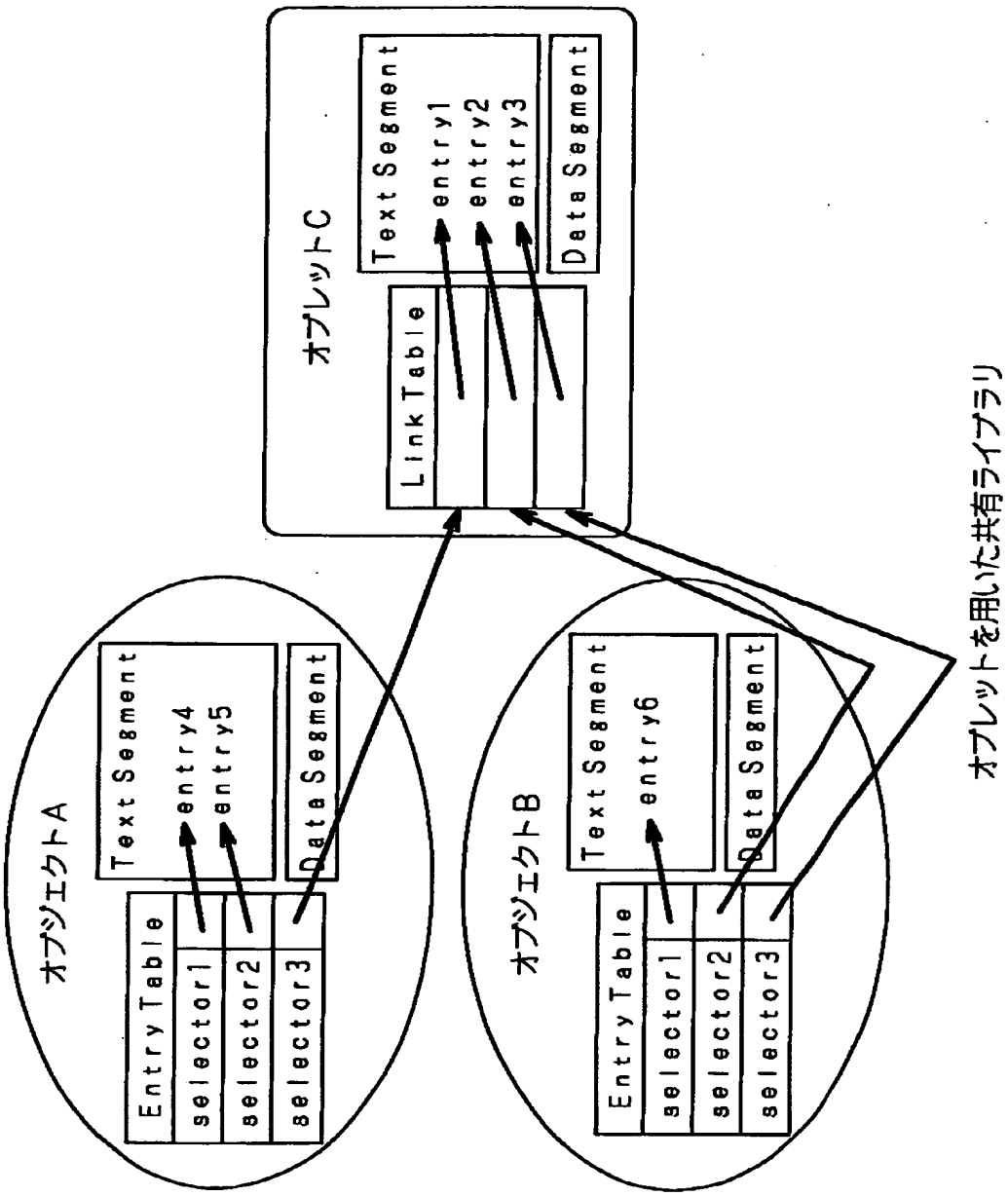
エントリテーブル

【図７】

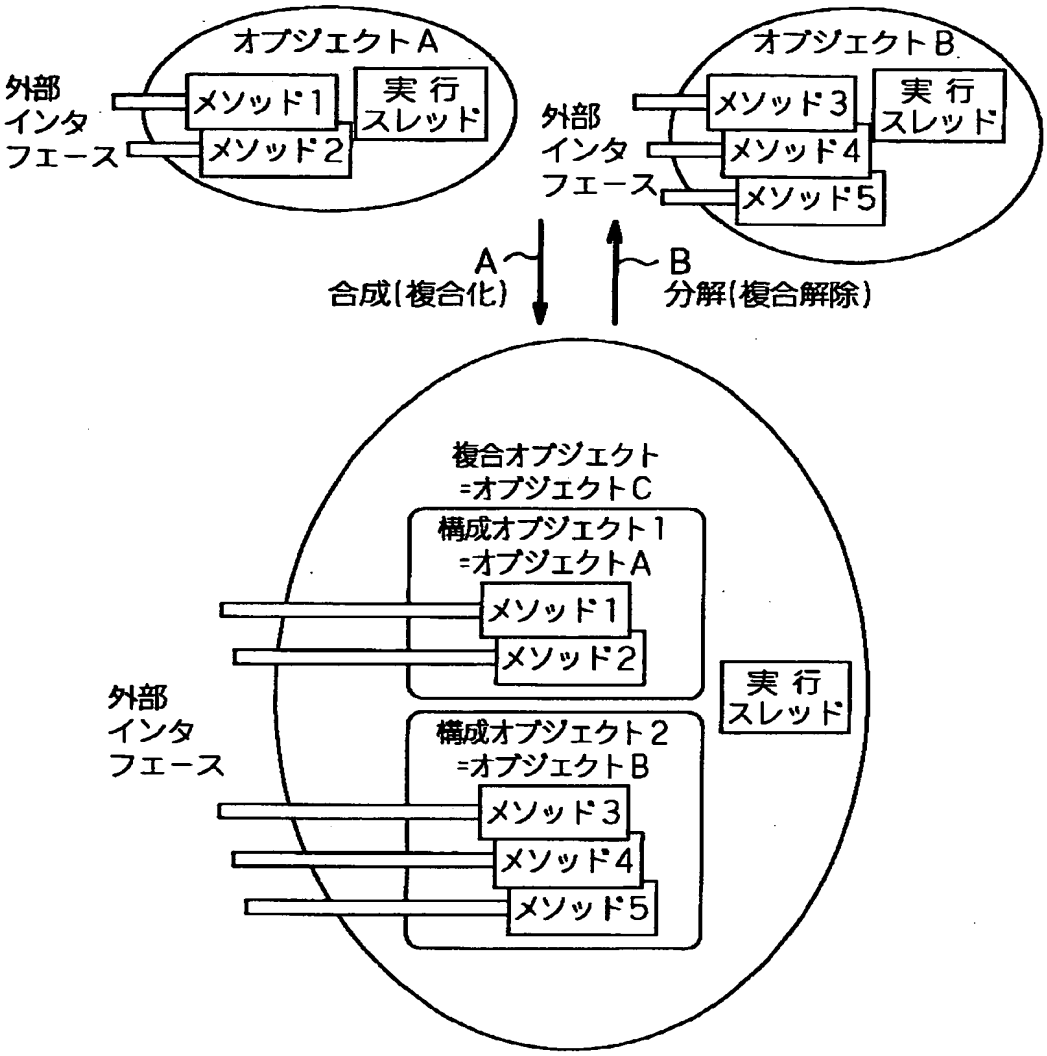


オブレットの構造

【図８】

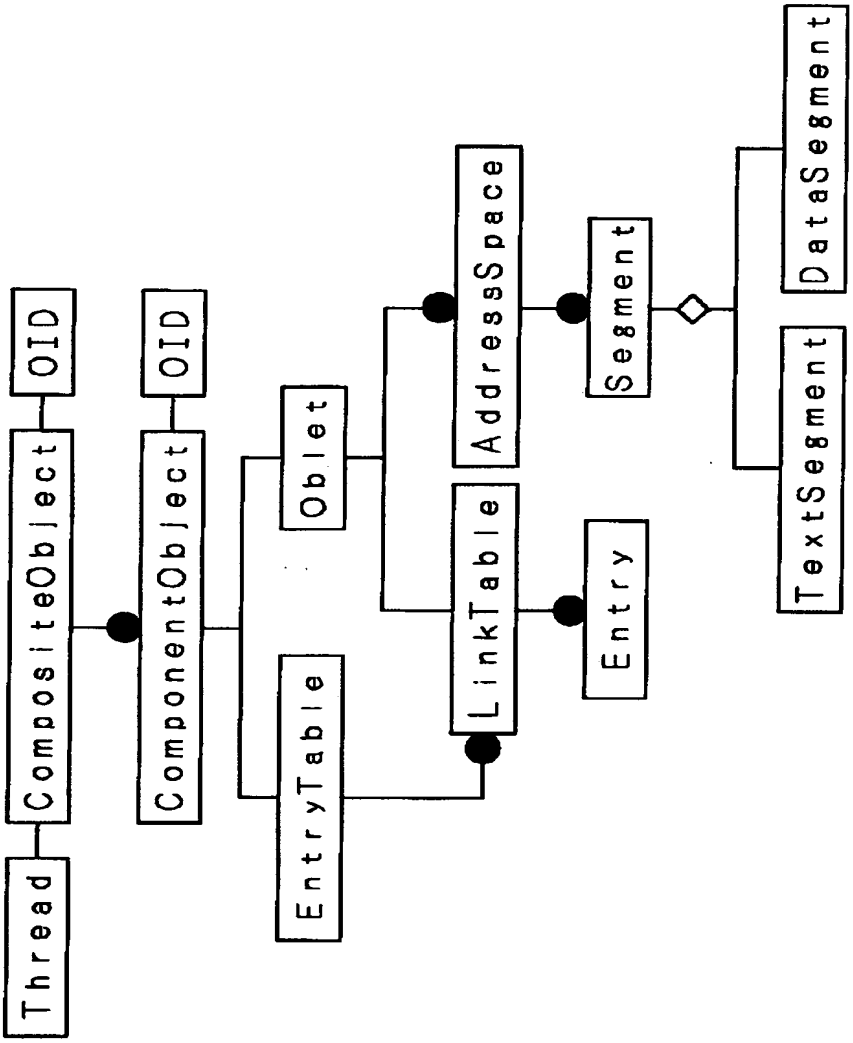

オブレットC
Text Segment
entry1
entry2
entry3
Data Segment
Link Table
オブジェクトA
Text Segment
entry4
entry5
Data Segment
Entry Table
selector1
selector2
selector3
オブジェクトB
Text Segment
entry6
Data Segment
Entry Table
selector1
selector2
selector3
オブレットを用いた共有ライブラリ

【図９】


オブジェクトA
外部
インタ
フェース
メソッド1
メソッド2
実 行
スレッド
オブジェクトB
外部
インタ
フェース
メソッド3
メソッド4
メソッド5
実 行
スレッド
A
合成(複合化)
B
分解(複合解除)
複合オブジェクト
=オブジェクトC
構成オブジェクト1
=オブジェクトA
メソッド1
メソッド2
外部
インタ
フェース
実 行
スレッド
構成オブジェクト2
=オブジェクトB
メソッド3
メソッド4
メソッド5

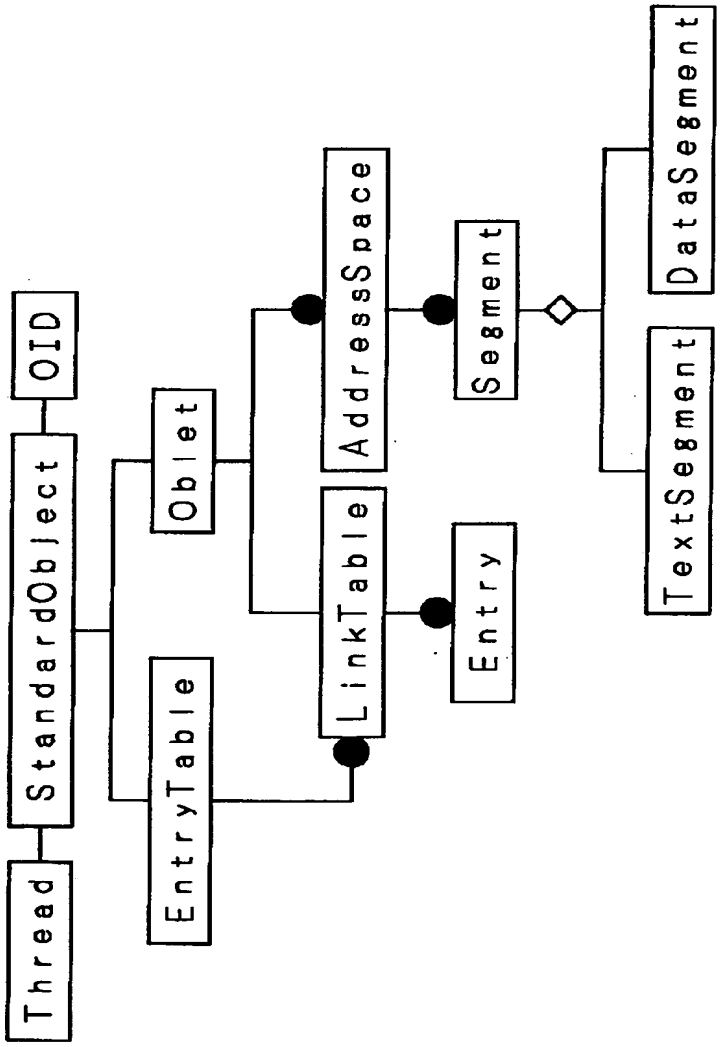
複合オブジェクト

【図１０】



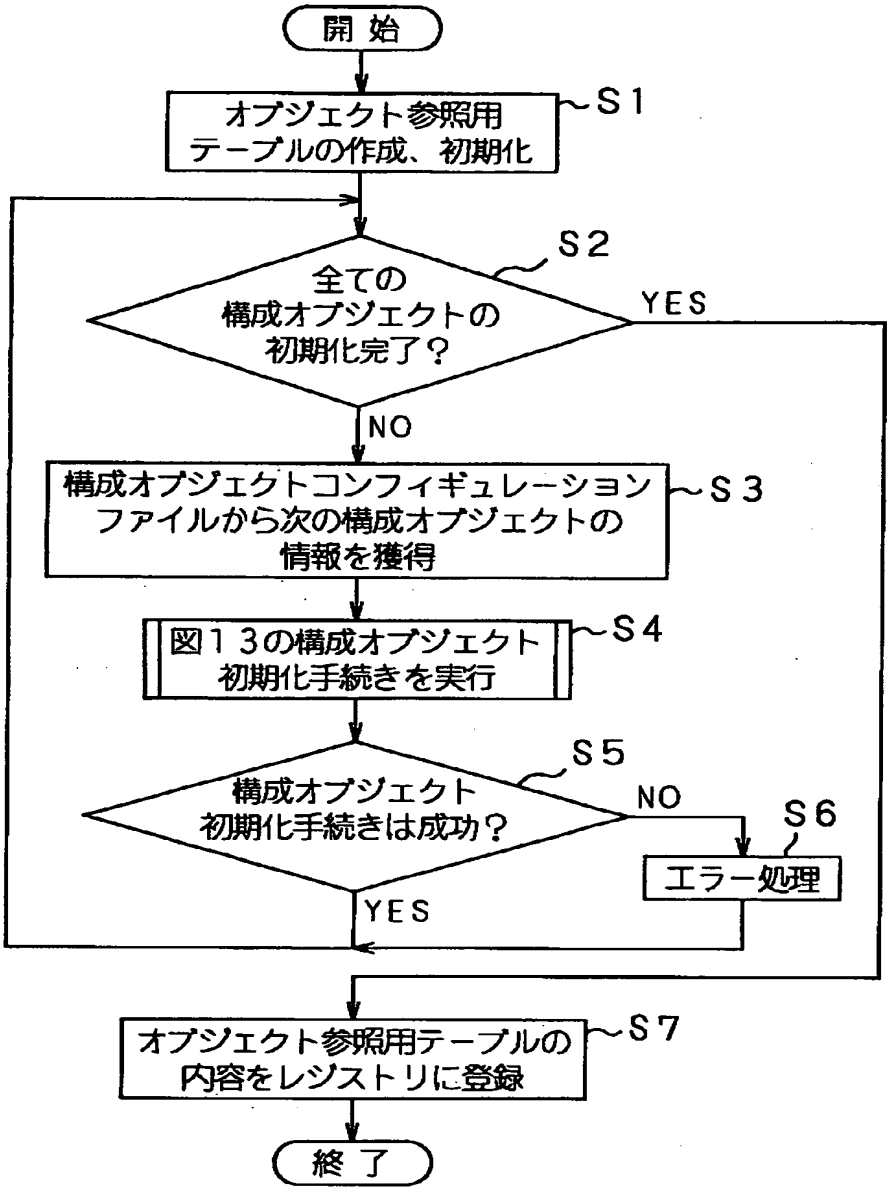
複合オブジェクトの構成

【図１１】



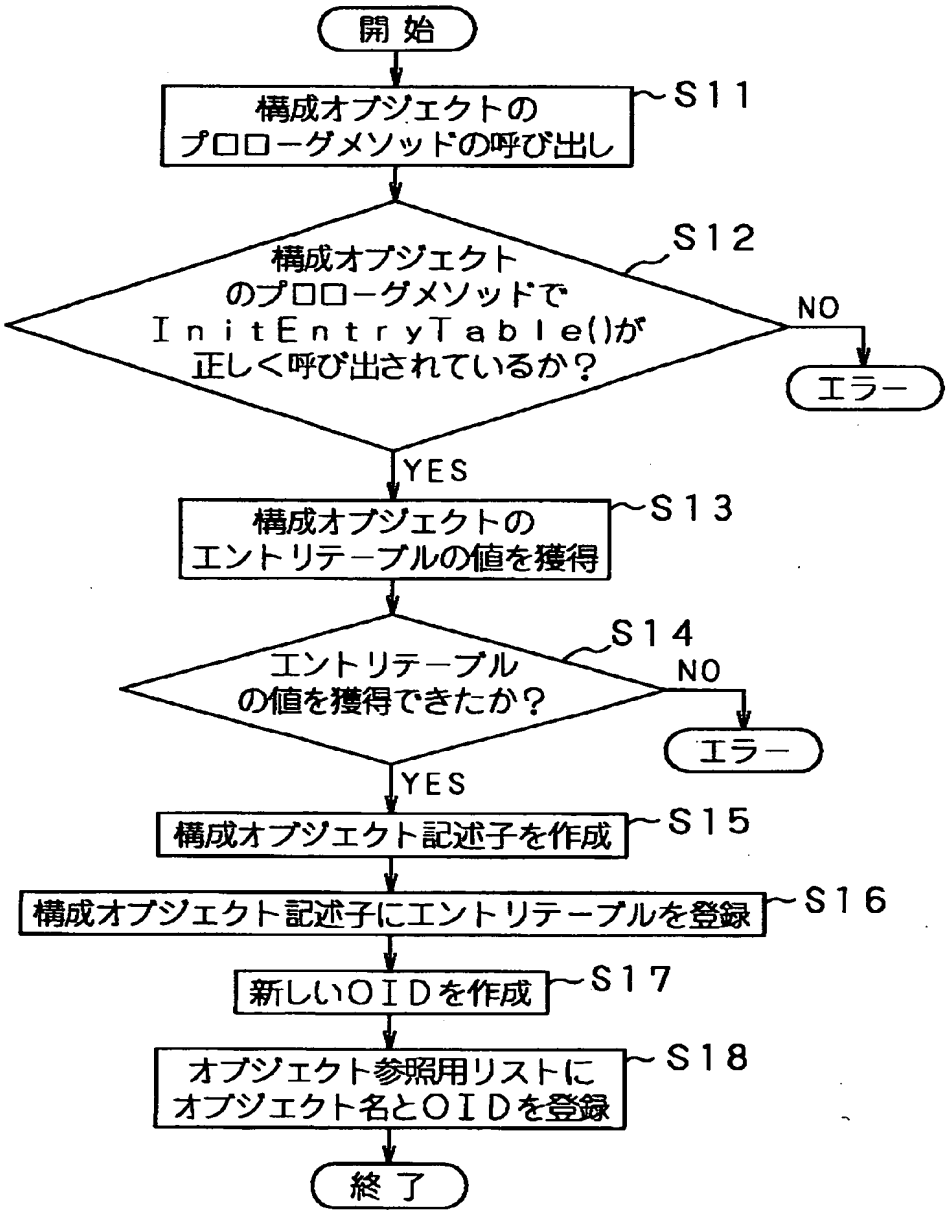
標準オブジェクトの構成

【図１２】



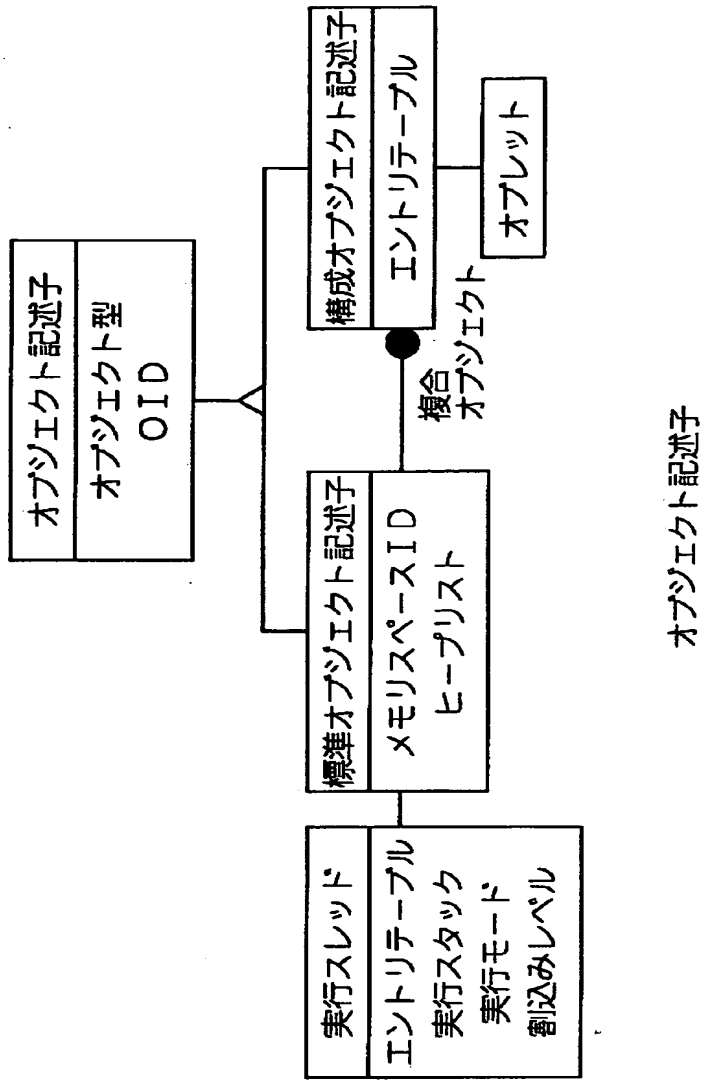
複合オブジェクトの初期化手続き

【図１３】

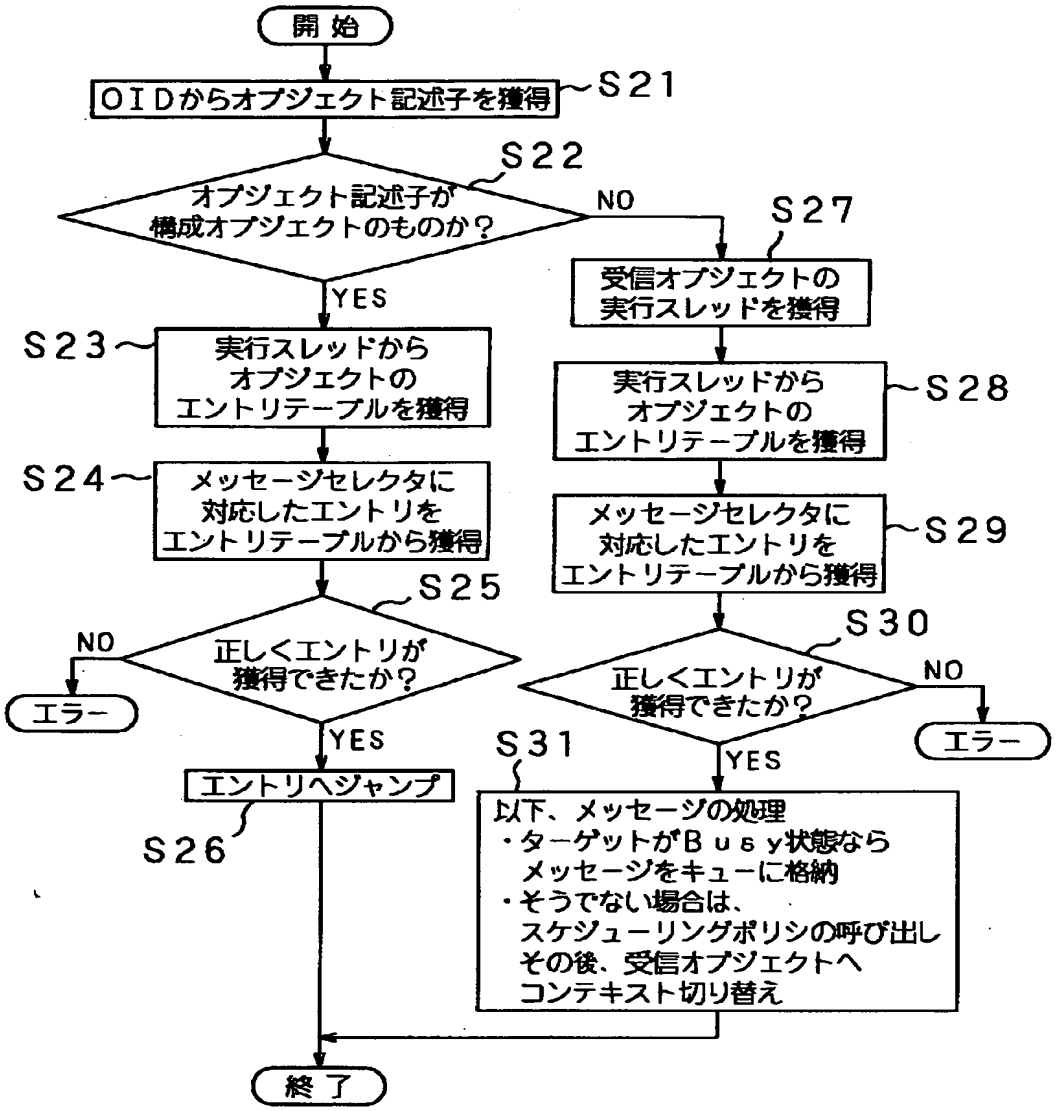


構成オブジェクトの初期化手続き

【図１４】

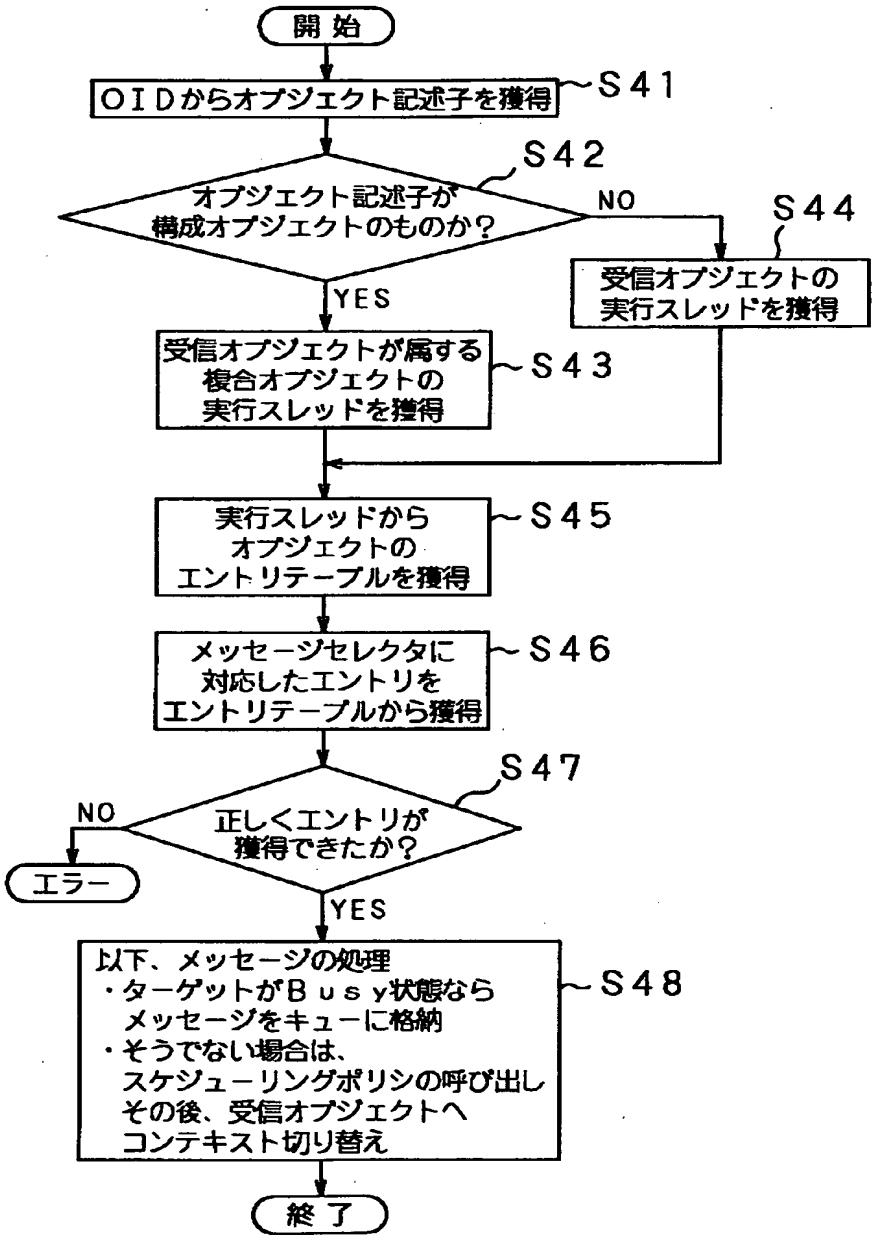

オブジェクト記述子
オブジェクト型
ＯＩＤ
構成オブジェクト記述子
エントリテーブル
オブレット
複合
オブジェクト
標準オブジェクト記述子
メモリスペースＩＤ
ヒープリスト
実行スレッド
エントリテーブル
実行スタック
実行モード
割込みレベル
オブジェクト記述子

【図１５】



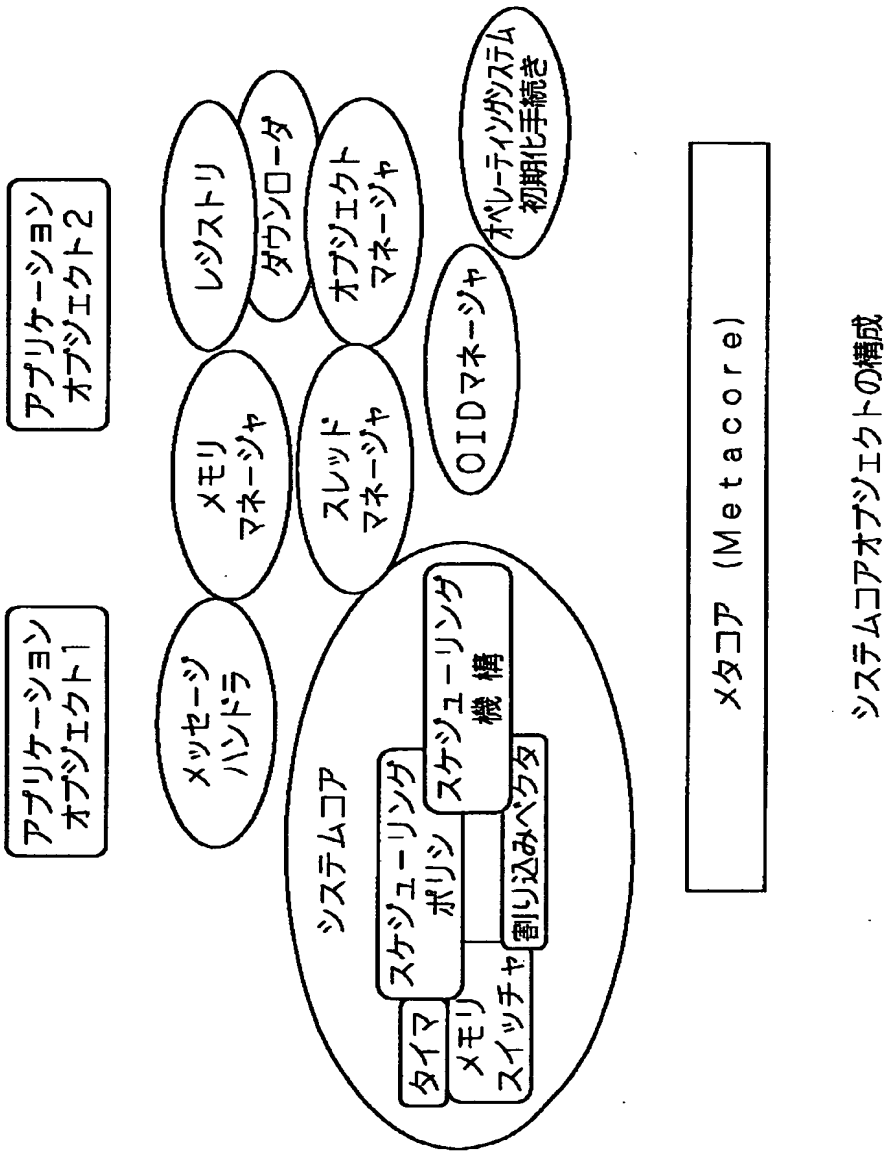
構成オブジェクトからのメッセージ送信

【図１６】



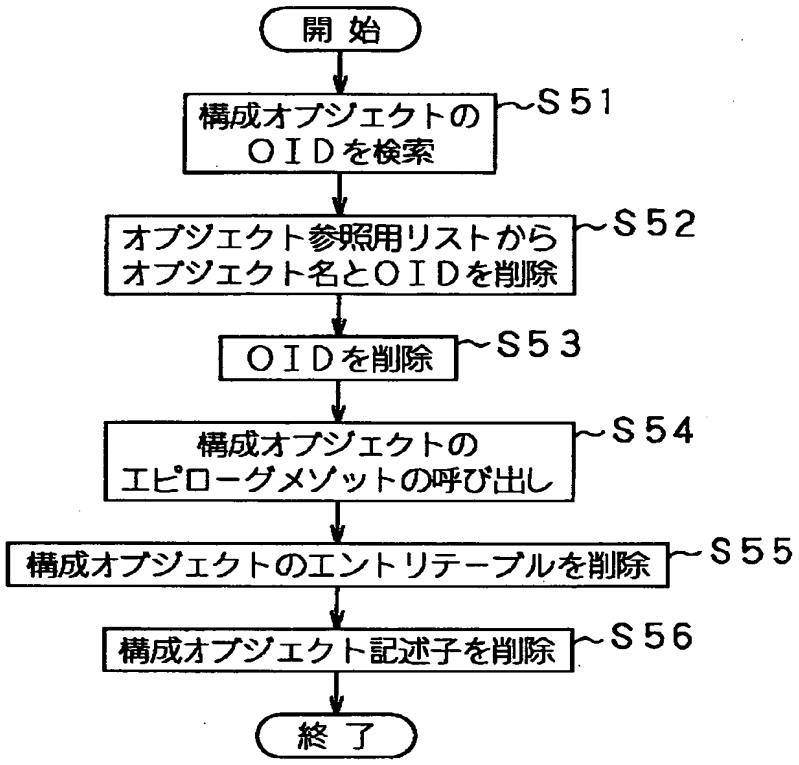
構成オブジェクトからのメッセージ送信

【図１７】



システムコアオブジェクトの構成

【図１８】



構成オブジェクトの削除手続き

【書類名】　要約書

【要約】

【課題】　オブジェクト指向オペレーティングシステムを採用したシステムにおいて、システムの柔軟性を保ちつつ、システム全体の実行性能を向上させる。

【解決手段】　オブジェクト間でメッセージ通信を行うオブジェクトを、１つ以上の構成オブジェクトから構成される複合オブジェクトと、複合オブジェクト以外のオブジェクトである標準オブジェクトとのいずれかにより構成する。そして、任意のオブジェクトから標準オブジェクト及び構成オブジェクトを参照できるように、各標準オブジェクト及び各構成オブジェクトに識別子を付す。また、複合オブジェクトについては、１つの複合オブジェクトを１つの実行スレッドによって実行し、その実行スレッドを複合オブジェクトを構成する各構成オブジェクトによって共有させる。

【選択図】　図９

出　願　人　履　歴　情　報

識別番号　　〔０００００２１８５〕

１．変更年月日　　１９９０年　８月３０日
［変更理由］　新規登録
住　所　　東京都品川区北品川６丁目７番３５号
氏　名　　ソニー株式会社